VŠB – Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

# Detekce a evoluce komunit v komplexních sítích

# Detection and Evolution of Communities in Complex Networks

2018

Bc. Martin Buček

VŠB - Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

# Zadání diplomové práce

Student: **Bc. Martin Buček**

Studijní program: N2647 Informační a komunikační technologie

Studijní obor: 2612T025 Informatika a výpočetní technika

Téma: Detekce a evoluce komunit v komplexních sítích
Detection and Evolution of Communities in Complex Networks

Jazyk vypracování: čeština

Zásady pro vypracování:

Cílem práce je prostudovat metody pro detekci komunit v komplexních dynamických sítích, vybrané metody implementovat a porovnat evoluce různě detekovaných komunit.

1. Prostudujte problematiku detekce komunit v dynamických sítích a jejich evoluce.
2. Vyberte vhodné datové kolekce.
3. Implemenujte vybrané (min. 4) algoritmy pro detekci komunit.
4. Experimentujte s evolucí komunit.
5. Vhodně zvolte reprezentaci získaných výsledků (případně vizualizujte pomocí programu Gephi nebo Netgram).

Seznam doporučené odborné literatury:

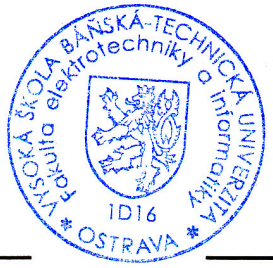
[1] Hartmann, Tanja, Andrea Kappes, and Dorothea Wagner. "Clustering evolving networks." Algorithm Engineering. Springer International Publishing, 280-329 (2016).
[2] Greene, D., Doyle, D. & Cunningham, P. Tracking the evolution of communities in dynamic social networks. Proceedings - 2010 International Conference on Advances in Social Network Analysis and Mining, ASONAM 2010 176-183 (2010).
[3] Mall, Raghvendra, Rocco Langone, and Johan AK Suykens. "Netgram: Visualizing Communities in Evolving Networks." PloS one 10.9 (2015).
A další literatura dle pokynů vedoucího práce.

Formální náležitosti a rozsah diplomové práce stanoví pokyny pro vypracování zveřejněné na webových stránkách fakulty.

Vedoucí diplomové práce: **Mgr. Pavla Dráždilová, Ph.D.**

Datum zadání: 01.09.2017

Datum odevzdání: 30.04.2018




doc. Ing. Jan Platoš, Ph.D.
*vedoucí katedry*

prof. Ing. Pavel Brandštetter, CSc.
*děkan fakulty*

Prohlašuji, že jsem tuto diplomovou práci vypracoval samostatně. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

V Ostravě 30. dubna 2018 .....

Rád bych na tomto místě poděkoval paní Mgr. Pavle Dráždilové, Ph.D. za její cenné rady poskytnuté během vypracovávání této diplomové práce.

**Abstrakt**

Náplní práce bylo implementovat algoritmy pro lokální detekci komunit a následně popsat jejich evoluční vývoj v časových řezech dynamických komplexních sítí. Pro tuto práci byla vybrána syntetická dynamická síť, která je výsledkem optimalizačního SOMA algoritmu a část spoluautorské sítě DBLP v časových řezech 2010 až 2015. Pro detekci lokálních komunit byly vybrány čtyři převzaté algoritmy. Některé z těchto algoritmů jsou parametrizovatelné, což dává uživateli možnost do určité míry ovlivňovat výslednou komunitu. Tato práce využívá pro detekci evolučních událostí jeden převzatý algoritmus a jeden vlastní, který byl navržen s ohledem na různorodost dat. Oba tyto algoritmy mají množinu nastavitelných parametrů. Tato práce obsahuje výsledky experimentů s detekčními i evolučními algoritmy a také popisuje jejich rozdíly.

**Klíčová slova**: analýza dat, teorie grafů, dynamické sítě, detekce komunit, evoluce komunit, diplomová práce

**Abstract**

This master's thesis focuses on the implementation of algorithms for local community detection and the subsequent description of their evolution process in time frames of dynamic complex networks. The thesis works with the synthetic dynamic network that results from the optimizing SOMA algorithm and with co-author dynamic network DBLP in the time frames from 2010 to 2015. For the local community detection, four adapted algorithms were selected. Some of these algorithms can be parameterized, thus enabling the user to partially influence the resulting community structure. For the detection of evolution events, one adapted algorithm is used, and one original is defined, with respect to data variability. Both these algorithms have the set of input parameters. The thesis summarizes the experiment results of using detection and evolution algorithms and describes their differences as well.

**Key Words**: data analysis, graph theory, dynamic networks, community detection, community evolution, master thesis

# Obsah

# Seznam použitých zkratek a symbolů

| | | |
|---|---|---|
| CPM | – | Clique Percolation Method |
| CSV | – | Comma-Separated Values |
| DBLP | – | Digital Bibliography & Library Project |
| GEXF | – | Graph Exchange XML Format |
| LOA | – | Local Optimal Algorithm |
| SOMA | – | Self Organizing Migrating Algorithm |
| SQL | – | Structured Query Language |
| WWW | – | World Wide Web |

# Seznam obrázků

# Seznam tabulek

# 1 Úvod

Tato práce se zaměřuje na detekci evolučních událostí v dynamických komplexních sítích. Pro sledování vývoje jsme se soustředili na evoluci komunit. V oblasti analýzy síťových dat již bylo představeno mnoho algoritmů pro detekci komunit. Všechny algoritmy pro detekci komunit, které byly pro tuto práci implementovány, vyžadují specifikaci počátečního vrcholu, který se nalézá v časových řezech dynamické sítě. Toto byl důvod, proč jsme se soustředili na algoritmy pro lokální detekci komunit.

V druhé kapitole jsou popsány základní pojmy, o které se tato práce práce opírá. Jsou zde uvedeny vlastnosti komplexních sítí a různé pohledy na definici komunit. V další kapitole jsou popsány základní přístupy k detekci komunit v sítích. V této kapitole je také uveden detailní popis konkrétních detekčních algoritmů, které byly vybrány pro implementaci. Ve čtvrté kapitole je definována evoluce komunit a popis jednotlivých evolučních událostí, kterými tato práce popisuje evoluční proces komunit. V této části jsou popsány dvě metody pro detekci evolučních událostí v dynamické síti. První metoda byla převzata z [25] a druhá byla navržena s ohledem na různorodost dat.

Předposlední kapitola popisuje implementaci aplikace. V této části je také uveden popis datových kolekcí, nad kterými byly prováděny experimenty. V poslední kapitole jsou popsány a vizualizovány výsledky experimentů s detekčními a evolučními algoritmy. Tato kapitola obsahuje jak porovnání algoritmů pro detekci komunit, tak i algoritmů pro detekci evolučních událostí. Závěr shrnuje dosažené výsledky a popisuje možné rozšíření této práce do budoucna.

# 2 Základní pojmy

V této kapitole jsou vysvětleny vybrané základní pojmy související s detekcí a evolucí komunit v komplexních sítích. Použité pojmy z oblasti teorie grafů jsou převzaty z [1].

## 2.1 Komplexní síť

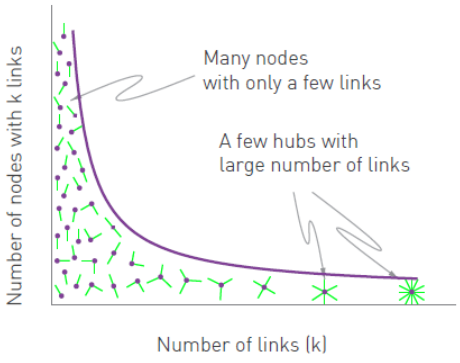
Komplexní síť je definována jako graf s netriviálními strukturálními vlastnostmi [2]. Mezi nejzákladnější způsob měření můžeme považovat počet vrcholů v grafu. Jinými slovy čím větší počet vrcholů graf má, tím je považován za komplexnější. Mezi další netriviální strukturální vlastnosti, které mohou být použity pro způsoby měření komplexity sítí, patří například vysoký shlukovací koeficient, koeficient asortativity, výskyt reciprocity u orientovaných grafů a mnohé další. Mnoho biologických, technologických a sociálních sítí vykazuje tyto vlastnosti, a proto jsou řazeny mezi komplexní. Náhodně generované grafy tyto charakteristiky nevykazují.

### 2.1.1 Bezškálové (Scale-free) sítě

Komplexní sítě, které mají charakteristickou distribuci stupňů vrcholů jsou označeny jako bezškálové. Pro bezškálové sítě platí, že jejich distribuce stupně vrcholů se řídí tzv. power-law distribucí [3].

$$P(k) \sim k^{-\gamma} \tag{1}$$

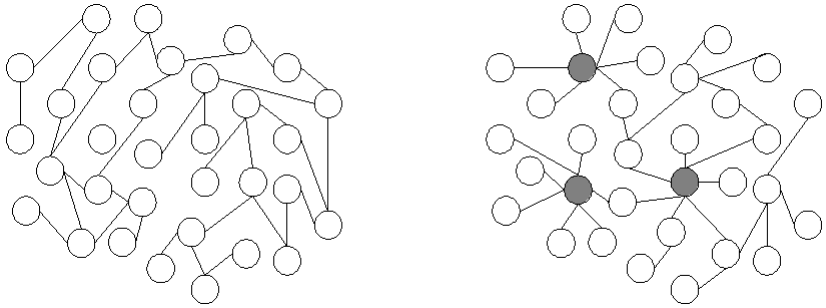
V rovnici 1 můžeme vidět definici power-law. $P(k)$ je pravděpodobnost, že vrchol má $k$ hran a $\gamma$ je reálný koeficient. Experimentálně bylo zjištěno, že u mnohých reálných sítí se hodnota $\gamma$ pohybuje v intervalu $\langle 2; 3 \rangle$, či někde poblíž. [3]. Pro orientované sítě lze ještě rozlišovat mezi $\gamma_{in}$ a $\gamma_{out}$ [3].




Obrázek 1: Power-law distribuce Zdroj: *NETWORK SCIENCE THE SCALE-FREE PROPERTY* [3]

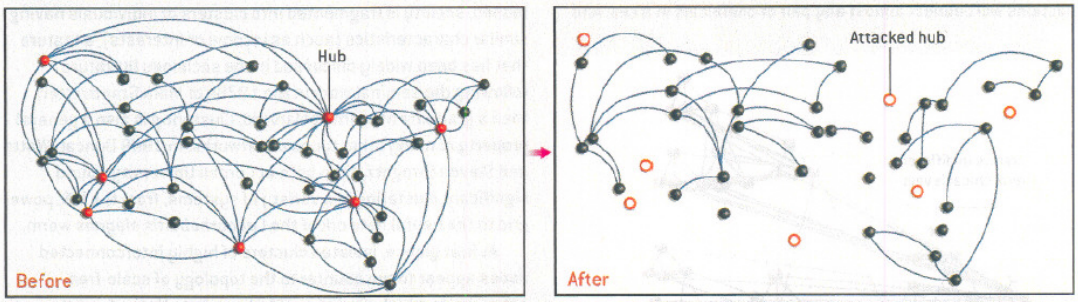
Bezškálové sítě obsahují podmnožinu vrcholů, které se nazývají *huby* a jsou charakteristické spojením s velkým množstvím jiných vrcholů v síti [4]. Jde o centrální vrcholy s vysokým stupněm. Většina vrcholů v síti má malý stupeň. Tento jev popisuje obrázek 1. V porovnání

s náhodně generovanými sítěmi je to velký rozdíl. Na obrázku 2 můžeme vidět porovnání struktury náhodně generované sítě (vlevo) a bezškálové sítě (vpravo). Tmavě vybarvené vrcholy ve bezškálové síti jsou výše zmíněné huby.



Obrázek 2: Porovnání náhodně generovaných sítí s bezškálovými sítěmi Zdroj: *What is a complex graph?* [2]

Bezškálové sítě mají z hlediska struktury další zajímavé vlastnosti. Jsou odolné vůči výpadkům, ale jsou velice zranitelné vůči koordinovaným útokům. Pro lepší pochopení si můžeme takovou síť představit jako počítačovou síť, kde vrcholy reprezentují síťové prvky a hrany reprezentují spojení v síti (optické, kabelové apod.). Tmavě vybarvené vrcholy by byly routery. V případě náhodného výpadku některého ze síťových prvků je velmi malá pravděpodobnost, že by šlo zrovna o router (hub), neboť v celé síti je jich velmi málo. V případě koordinovaného útoku útočníka, který by dokázal identifikovat v celé topologii sítě routery, by šlo o velký problém. Pokud by útočník vyřadil některý z hubů, většina sítě by se strukturálně rozpadla z důvodu vysokého stupně vrcholu. Příklad útoku na huby v bezškálové síti můžeme vidět na obrázku 3.




Obrázek 3: Ilustrace dopadu útoku na huby v bezškálové síti Zdroj: *Scale-Free Networks* [4]

### 2.1.2 Sítě malého světa

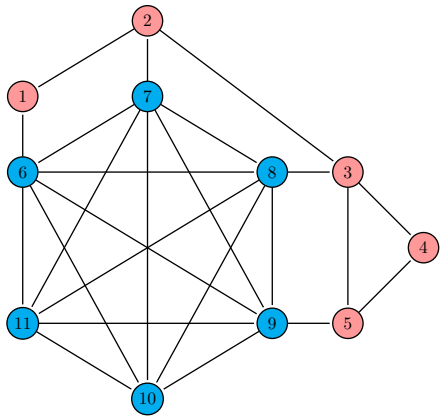
Komplexní sítě jsou také známé svou další charakteristickou vlastností, která se nazývá sítě malého světa (small world). Tato vlastnost se týká vzdálenosti mezi dvěma libovolnými vrcholy v komplexní síti. Experimentálně bylo zjištěno, že tato vzdálenost je i ve velmi rozsáhlých sítích malá.

V roce 1967 provedl psycholog Stanley Milgram sociální experiment, který se týkal této problematiky. Milgram rozeslal dopisy svým známým, kteří je měli za úkol předat svým známým takovým způsobem, aby se dopisy dostaly do cílového města Boston. Milgram zjistil, že průměrný počet předání dopisů mezi jednotlivými známými byl šest [5]. Tento jev je také nazýván šest stupňů odloučení. V minulosti byl proveden podobný experiment nad sítí webových stránek WWW (World Wide Web), službou poskytovanou v rámci internetu. V rámci tohoto experimentu bylo zjištěno, že průměrná vzdálenost mezi dvěma libovolně vybranými dokumenty v síti WWW je rovna číslu 19 [6].

## 2.2 Komunita

Abychom mohli vůbec začít detekovat komunity a sledovat jejich vývoj v dynamických sítích, je třeba nejprve definovat co je komunita a jaká je její struktura. V teorii sítí neexistuje zcela jednotná definice komunity, která by byla podložena nějakým matematickým aparátem a přijatá odbornou veřejností.

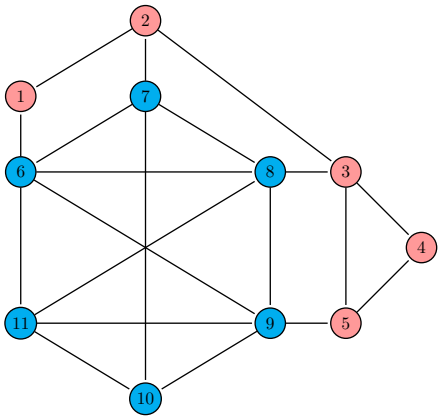
První z možností definicí komunit, která se nabízí, je definice pomocí tzv. úplného nebo-li kompletního grafu [7]. Komunita je definována jako podgraf isomorfní s kompletním grafem. Kompletní graf je tvořen $n$ vrcholy, kde každý je spojen hranami se zbývajícími $n-1$ vrcholy. Tomuto typu grafu se také říká n-klika, kde $n$ je počet vrcholů daného grafu. Značí se zkráceně $K_n$. Příklad komunity jako kompletního grafu můžeme vidět na obrázku 4, kde v tomto konkrétním případě tvoří $K_6$ (modrý podgraf).




Obrázek 4: Komunita definována jako úplný nebo-li kompletní podgraf $K_6$

Jedna z metod pro detekci komunit využívá nalezených úplných podgrafů. Metoda se jmenuje CPM (Clique percolation method) a v této práci není použita.

Komunita je také někdy definována jako podgraf, který je spojen se zbytkem sítě pouze několika hranami, zatímco uvnitř samotného podgrafu jsou jeho vrcholy hustě spojeny hranami [7]. Tato definice je v praxi často používaná. Existují metody, které pro detekci těchto struktur porovnávají počty hran uvnitř komunity s počtem hran přilehlých. Tento poměr se nazývá lokální modularita. Tento poměr je využit u detekčních algoritmů v této práci, jakou jsou například




Obrázek 5: Komunita definována jako propojenější část grafu

algoritmy Clauset (viz kapitola 3.3.2) nebo LOA (viz kapitola 3.3.3). Komunita tedy nemusí být definována striktně jako úplný podgraf. Na obrázku 5 můžeme vidět podgraf, který dle této definice tvoří komunitu. Jedná se o modrou propojenější část grafu.

# 3 Detekce komunit

V této kapitole je popsána problematika základního rozdělení přístupů k detekci komunit a následně detailní popis jednotlivých vybraných algoritmů.

## 3.1 Algoritmy s globálním přístupem

Prvním přístupem k detekci komunit jsou algoritmy s globálním přístupem. Tyto algoritmy pracují s celým grafem a jejich výstupem je rozdělení celého grafu do komunit. Výhodou je znalost struktury celé sítě i jejich komunit. Tento přístup je méně vhodný pro velmi rozsáhlé grafy. Pro větší sítě s tisíci uzly a řádově výše, bude časová i paměťová složitost velmi narůstat. Jako extrémní příklad můžeme uvést sociálně sítě typu Facebook nebo také internet, kde se počet vrcholů pohybuje řádově ve stovkách miliónů. Některé přístupy budou popsány v dalších podkapitolách.

### 3.1.1 Hierarchické shlukování

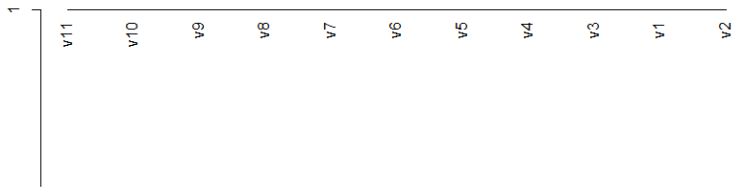
Mezi první zmíněné globální přístupy k detekci komunit patří hierarchické shlukování. Tato metoda pracuje se vzdáleností dvou vrcholů v síti. Jedná se o posloupnost vnořených rozkladů, která začíná triviálním rozkladem, kde každý vrchol v síti je brán sám o sobě jako jednoprvkový shluk. Končí triviálním rozkladem s jedním globálním shlukem obsahujícím všechny vrcholy [9]. Podle směru postupu při shlukování dělíme metody hierarchického shlukování na aglomerativní a divizivní.

Graficky lze hierarchické shlukování znázornit dendrogramem. Dendrogram je binární strom, kde každý list v tomto stromu představuje jednoprvkový shluk. Horizontální řezy dendrogramem představují rozklady v posloupnosti rozkladů a vertikální směr představuje vzdálenost mezi jednotlivými shluky [9]. Příklady dendrogramů můžeme vidět na obrázcích 6, 7 a 8. Tyto dendrogramy byly vytvořeny třemi metodami aglomerativního shlukování nad grafem, který je na obrázku 5. Pro jejich tvorbu byla vypočtena matice vzdáleností pomocí ceny nejkratší cesty mezi vrcholy. Následně byly dendrogramy s pomocí této matice vygenerovány v jazyce R.

#### 3.1.1.1 Aglomerativní shlukování

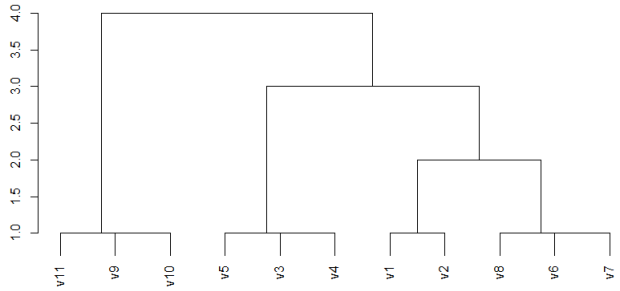
Algoritmus aglomerativního shlukování postupuje tzv. zdola nahoru. Se vstupní množinou objektů (vrcholů) pracuje jako s jednoprvkovými shluky (singletony). Algoritmus vybere dva jednoprvkové shluky vrcholů, které si jsou nejblíže a ty sloučí do jednoho většího shluku. Jako metriku používá nejkratší cestu mezi dvěma prvky $u, v$. V každém dalším kroku algoritmus vytváří z nejbližších shluků nižšího stupně rozkladu nový shluk. Nyní se pojďme podívat na jakých principech jsou založeny metriky počítající vzdálenost mezi shluky.

- **Metoda nejbližšího souseda** - vzdálenost mezi dvěma shluky se počítá na základě minimální vzdálenosti $d_{ij}$, kde objekt $i$ patří do jednoho shluku a objekt $j$ do druhého. Tato metoda vytvoří pro souvislý graf dendrogram na jedné úrovni.



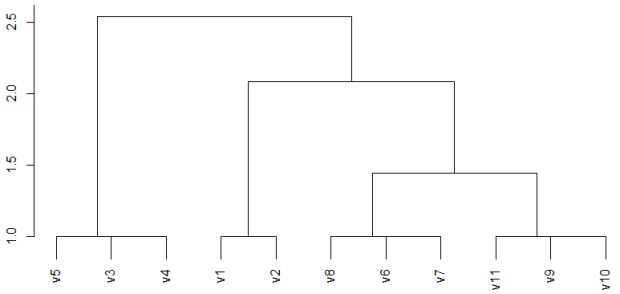

Obrázek 6: Dendrogram vytvořený metodou nejbližšího souseda nad grafem z obrázku 5

- **Metoda nejvzdálenějšího souseda** - vzdálenost mezi dvěma shluky se počítá na základě maximální vzdálenosti $d_{ij}$, kde objekt $i$ patří do jednoho shluku a objekt $j$ do druhého.




Obrázek 7: Dendrogram vytvořený metodou nejvzdálenějšího souseda nad grafem z obrázku 5

- **Metoda průměrné vazby** - vzdálenost mezi dvěma shluky se počítá jako průměr vzdáleností mezi každými dvěma objekty patřícími do dvou různých shluků. Tato metoda není příliš citlivá na statistické odchylky v datech.




Obrázek 8: Dendrogram vytvořený metodou průměrné vazby nad grafem z obrázku 5

U dendrogramu na obrázku 8 můžeme vidět, že pokud provedeme horizontální řez ve vzdálenosti 1.5, dostaneme shluk odpovídající komunitě znázorněné na obrázku 5. V tabulce 1 můžeme vidět

bližší pohled na výpočet vzdáleností pro jednotlivé shlukovací metody, kde $X, Y$ je označení pro shluky a $x, y$ jsou takové uzly, že platí $x \in X$ a $y \in Y$.

| **Shlukovací algoritmus** | **Vzorec pro výpočet vzdálenosti** |
|---|---|
| Metoda nejbližšího souseda | $d_{SL}(X,Y) = min_{x \in X, y \in Y} d(x,y)$ |
| Metoda nejvzdálenějšího souseda | $d_{CL}(X,Y) = max_{x \in X, y \in Y} d(x,y)$ |
| Metoda průměrné vazby | $d_{AL} = \frac{1}{\lvert X \rvert \cdot \lvert Y \rvert} \sum_{\forall x \in X} \sum_{\forall y \in Y} d(x,y)$ |

Tabulka 1: Výpočet vzdáleností mezi shluky

#### 3.1.1.2 Divizivní shlukování

Algoritmus divizivního shlukování postupuje opačně než v předešlém případě, tzv. shora dolů. Myšlenka tohoto algoritmu tkví v identifikaci hran, které spojují dva odlišné shluky. Pokud všechny tyto hrany odstraníme, rozdělíme jeden shluk na dva [7]. Divizivní shlukování je časově náročné, protože existuje $\mathcal{O}(2^n)$ možností, jak rozdělit množinu uzlů do dvou shluků [8].

### 3.1.2 Nehierarchické metody shlukování

Tato podkapitola popisuje princip nehierarchických metod shlukování. Tyto metody nevytvářejí hierarchickou strukturu, rozkládají danou množinu do podmnožin dle předem daného kritéria.

#### 3.1.2.1 Spektrální shlukování

Spektrální shlukování pracuje s maticí podobnosti, nikoliv s maticí vzdálenosti, jako tomu bylo v předešlém případě. Mějme množinu objektů $x_1$, $x_2$, $x_3$, ..., $x_n$. Dále definujme nad těmito objekty symetrickou a nezápornou podobnostní funkci:

$$Similarity(x_i, x_j) = Similarity(x_j, x_i) \geq 0, \forall i, j = 1, ..., n \tag{2}$$

Podobnost je v síti dána pomocí existence neorientovaných hran. Podobné uzly jsou spojené hranou, to znamená že místo matice podobnosti používáme matici sousednosti. Obecně tuto funkci můžeme použít na jakoukoliv množinu prvků i na graf. Spektrální shlukování používá k rozdělení množiny prvků do shluků vlastní vektory Laplaceovy matice. Nejprve transformuje původní množinu objektů na množinu bodů v prostoru. Koordináty těchto bodů jsou elementy vlastních vektorů. Shlukování těchto bodů dále probíhá klasickými známými algoritmy jako je třeba algoritmus k-means [7].

Spektrální shlukování pracuje s Laplaceovými maticemi. Používá jejich nenormalizovaný i normalizovaný tvar. Výpočet nenormalizovaného tvaru Laplacelovy matice můžeme vidět níže:

$$L = D - A, \tag{3}$$

kde $D$ je označení pro diagonální matici se stupni jednotlivých vrcholů a $A$ značí matici sousednosti. Díky vlastním vektorům Laplaceových matic může spektrální shlukování rozdělit množinu (graf) do požadovaných shluků [7].

## 3.2 Algoritmy s lokálním přístupem

Jiným přístupem k detekci komunit se zabývají algoritmy s lokálním přístupem. Tento přístup nepracuje s celou sítí, ale s počátečním vrcholem, který je libovolně volitelný. Algoritmus s lokálním přístupem iterativně expanduje od vybraného vrcholu k nejbližším sousedům, dokud není splněna nějaká ukončovací podmínka specifická pro daný algoritmus. Výstupem algoritmu je nalezená komunita obsahující počáteční vrchol. Výběr vrcholu výrazně ovlivní mohutnost výsledné komunity.

Výhoda oproti algoritmům s globálním přístupem je v tom, že algoritmus nepracuje s celou sítí, tudíž paměťová i časová složitost je výrazně nižší. Na druhou stranu ve výsledku získáme informaci pouze o jedné komunitě. Množství získaných informací je tedy menší.

V mnoha případech, zejména u rozsáhlých sítí, bývá problém zvolit vhodný počáteční vrchol. Volba počátečního vrcholu ovlivňuje výslednou komunitu.

## 3.3 Vybrané algoritmy pro lokální detekci komunit

Tato kapitola se zabývá podmnožinou algoritmů vybraných pro implementaci k detekci lokálních komunit v této práci. Algoritmy jsou popsány slovně z teoretického hlediska, ale i pseudokódem.

### 3.3.1 Algoritmus Shell

První z uvedených algoritmů pro detekci lokálních komunit se nazývá Shell (v překladu slupka). Celý běh algoritmu začíná v počátečním zvoleném vrcholu sítě. Postupně přistupuje k nejbližším sousedům počátečního vrcholu. Množině nově objevených sousedů budeme říkat slupka [10].

Algoritmus pracuje s dvěma důležitými mírami. Dle anglické literatury jsou tyto míry pojmenováno jako emerging degree a total emerging degree. Emerging degree je počet hran vedoucích z konkrétního vrcholu $i$ ve slupce $s$ do vrcholů v následující slupce $s$+1. Total emerging degree je definováno jako součet všech hran vedoucích ze všech vrcholů ve slupce $s$ do vrcholů ve slupce $s$+1 [10].

$k_i(j)$ = emerging degree uzlu i a j je označení pro počáteční vrchol shellu,

$K_j^s$ = total emerging degree slupky hloubky s a j je označení pro počáteční vrchol shellu,

$$\text{kde formálně: } K_j^s = \sum_{i \in s} k_i(j) \tag{4}$$

V počáteční fázi algoritmu (hloubka $s$=0), kdy známe pouze počáteční vrchol, bude total emerging degree roven stupni počátečního vrcholu.

$$K_j^0 = k_j \tag{5}$$

Nyní si definujme další důležitou míru, se kterou algoritmus pracuje. V anglické literatuře se jedná o change of total emerging degree [10].

$$\Delta K_j^s = \frac{K_j^s}{K_j^{s-1}} \tag{6}$$

Tato míra vyjadřuje poměr total emerging degree současné slupky s total emerging degree $s$-1 slupky. Využívá se pro ukončovací podmínku celého algoritmu. V každé iteraci algoritmu se nejprve naleznou nejbližší sousedé a poté se vypočte change total emerging. Následně se porovná s parametrem $\alpha$, který je uživatelský volitelný.

Pokud je poměr menší než parametr $\alpha$, algoritmus se následně ukončí. Do výsledné lokální komunity patří všechny vrcholy, které byly nalezeny expanzí algoritmu do hloubky $s$-1. Pokud je parametr naopak větší než parametr $\alpha$, algoritmus pokračuje dále ve svém expanzivním chování. Celý běh algoritmu můžeme popsat v následujících krocích[10].

1. Volba počátečního vrcholu $j$ algoritmu Shell.
2. Spuštění algoritmu Shell (hloubka $s$=0) v počátečním vrcholu. Přidáme vrchol $j$ do lokální komunity a vypočteme $K_j^0$.
3. V hloubce $s$=1 přidáme všechny sousedy vrcholu $j$ do dočasné kolekce a spočteme $K_j^s$.
4. Vypočteme $\Delta K_j^s$. Jestliže platí podmínka $\Delta K_j^s < \alpha$, algoritmus se automaticky ukončí a komunita byla nalezena.
5. V případě, že podmínka v kroku 4 není splněna, dojde k expanzi do další slupky a pokračuje opět krokem 4 dokud nenalezne výslednou komunitu.

Níže můžeme vidět celý pseudokód algoritmu Shell v algoritmu 1.

**Algoritmus 1:** Pseudokód pro algoritmus Shell

**input :** Graph $G$, seed $v_0$ and parameter $\alpha$
**output:** Community $C$ containing $v_0$

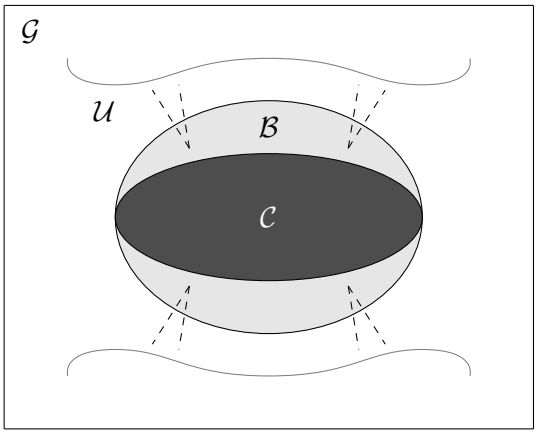
$K_s^{d-1} = 1$;
$C = \emptyset$;
$Q = \emptyset$;
$C.enqueue(v_0)$;
$K_s^d = emerging(Q, C, G(V, E))$;
$\Delta K_s^d = \frac{K_s^d}{K_s^{d-1}}$;
**while** $\Delta K_s^d > \alpha$ **do**
    $\Delta K_s^{d-1} = K_s^d$;
    **foreach** $q$ *in* $Q$ **do**
        $q = Q.dequeue()$;
        $C.enqueue(q)$;
        $Q.enqueue(neighbors(q))$;
    $K_s^d = emerging(Q, C, G(V, E))$;
    $\Delta K_s^d = \frac{K_s^d}{K_s^{d-1}}$;
**return** $C$;

Jak již bylo zmíněno výše, onen číselný parametr $\alpha$ je uživatelsky volitelný. Množství vrcholů v nalezené komunitě je silně závislé na zvolení tohoto parametru. Může nastat i situace, kdy lokální komunitu bude tvořit pouze počáteční zvolený vrchol. A to pouze v případě, že by daný vrchol měl emerging degree menší než je hodnota parametru $\alpha$. Taková komunita se nazývá v anglické literatuře *singleton community*. Je důležité hodnotu parametru volit s rozvahou. Podrobné výsledky experimentů ovlivněné na základě manipulace s tímto parametrem jsou popsány v kapitole experimentů 6.1.1, která se této problematice věnuje.

### 3.3.2 Algoritmus Clauset

Dalším algoritmem v oblasti lokálních přístupů k detekci komunit je Clauset. Z hlediska implementace se jedná o podstatně složitější algoritmus než předešlý Shell. Na obrázku 9 můžeme vidět definici struktury komunity dle algoritmu Clauset.

Jádro komunity označme jako $\mathcal{C}$ (core). V jádru se nachází všechny vrcholy, které nejsou spojeny žádnou hranou s nějakým vrcholem z neznámého okolí komunity. Jelikož je komunita propojena pro nás s neznámým okolím v síti, musíme definovat i dané okolí $\mathcal{U}$ (unknown). Dále definujme hraniční oblast komunity $\mathcal{B}$ (boundary). Do této oblasti patří všechny vrcholy, které jsou spojeny alespoň jednou hranou s nějakým vrcholem z neznámého okolí komunity $\mathcal{U}$. Tímto rozdělením získáme přesný tvar komunity. Clauset pracuje s velmi důležitou mírou, která se




Obrázek 9: Definice struktury komunity dle Clausetu

nazývá lokální modularita $R$[11, 12].

$$R = \frac{\sum_{ij} B_{ij}\delta(i,j)}{\sum_{ij} B_{ij}} = \frac{I}{T}, \tag{7}$$

kde hodnota funkce $\delta(i,j)$ je 1 v případě, že platí $v_i \in \mathcal{C}$ a zároveň $v_j \in \mathcal{B}$ nebo naopak. Jinak je hodnota $\delta(i,j) = 0$. $I$ ve vzorci reprezentuje počet hran, které mají jeden vrchol v jádru $\mathcal{C}$ a druhý v $\mathcal{B}$. Naopak $T$ reprezentuje počet hran, které mají aspoň jeden koncový vrchol v hraniční oblasti $\mathcal{B}$. Jinými slovy mohou tyto hrany vést buď do jádra nebo do neznámého okolí komunity. Tento poměr nám zajišťuje, že hodnoty lokální modularity budou nabývat hodnot v intervalu $0 \leq \mathrm{R} \leq 1$ [11], jelikož množina hran $I$ je podmnožinou množiny hran $T$. Modularita nám říká, zda je právě detekovaná komunita dostatečně silná nebo nikoliv.

Jako každý jiný algoritmus s lokálním přístupem, celý běh Clausetu začíná v počátečním vrcholu $v_0$, který jako první přidáme do kolekce naší lokální komunity $\mathcal{C}$.

**Algoritmus 2:** Pseudokód pro algoritmus Clauset

**input :** Graph $G$, seed $v_0$ and parameter $k$
**output:** Community $C$ containing $v_0$

```
Inicialization C = ∅, B = ∅, U = ∅;
Add v_0 to C, B and its neighbors to U;
while |C| < k do
    foreach v_j in U do
        compute ΔR_j
    find v_j such that ΔR_j is maximum;
    add that v_j to C;
    add all new neighbors of that v_j to U;
    update R and B;
return C;
```

Všechny přímé sousední vrcholy přidáme do okolí komunity $\mathcal{U}$. Jelikož je ze začátku známe pouze počáteční vrchol $v_0$ a jeho sousedy, kteří jsou všichni v neznámém okolí komunity, musíme zároveň přidat počáteční vrchol do hraniční oblasti komunity $\mathcal{B}$. Do hraniční oblasti patří vrcholy, které jsou spojeny hranou s neznámým okolím komunity $\mathcal{U}$.

Dále algoritmus přidá dočasně po jednom všechny vrcholy $v_j \in \mathcal{U}$ do komunity $\mathcal{C}$. Přidá se vždy jeden vrchol a pak algoritmus následně spočítá přírůstek lokální modularity $\Delta R_j$ nové dočasné komunity. Vrchol se poté odebere, tím dostaneme původní komunitu a celý proces se zase opakuje pro zbývající vrcholy v $\mathcal{U}$. Vrchol, který způsobí největší přírůstek lokální modularity se nakonec přidá do výsledné komunity. Dalším krokem algoritmu je přidání všech sousedů nově přidaného vrcholu $v_j$ do $\mathcal{U}$. Nepřidáváme ty sousedy, které jsou již zařazené v lokální komunitě. Posledním krokem algoritmu je aktualizace hraniční oblasti $\mathcal{B}$ a lokální modularity dle vzorce 7. Aktualizace hraniční oblasti komunity probíhá tím způsobem, že se ověří zda nově přidaný vrchol nemá vazby na některý z vrcholů v okolí komunity. Pokud tato podmínka platí, vrchol se přesune do hraniční oblasti $\mathcal{B}$. Přidáním nového vrcholu do komunity nebo hraniční oblasti se změní počet hran v množinách $I$ a $T$. Z toho důvodu se provede aktualizace hodnoty lokální modularity dle vzorce 7.

Clauset opakuje tento děj iterativně tak dlouho, dokud mohutnost celé množiny komunity nedosáhne hodnoty parametru $k$ nebo pokud nenalezne celou hledanou komunitu. Komunita je nalezena tehdy, když je na konci iterace $t$+1 po aktualizaci lokální modularity hodnota $R$ menší než v předešlé iteraci $t$.

V odstavci výše je zmíněn přírůstek lokální modularity $\Delta R_j$. Jeho výpočet, který je spjat s každým $v_j \in \mathcal{U}$ můžeme odvodit z následujícího vzorce 8 [11]:

$$\Delta R_j = \frac{x - Ry - z(1 - R)}{T - z + y}, \tag{8}$$

kde $x$ je počet hran, které patří do množiny $T$ a zároveň mají jeden koncový vrchol $v_j$. Koeficient $y$ je počet hran, které budou přidány do komunity zařazením vrcholu $v_j$ do komunity. Zároveň platí, že stupeň $v_j$ se dá vyjádřit jako součet $x$+$y$. Poslední koeficient $z$ vyjadřuje počet hran, které budou ubrány odebráním $v_j$ ze stávající komunity.

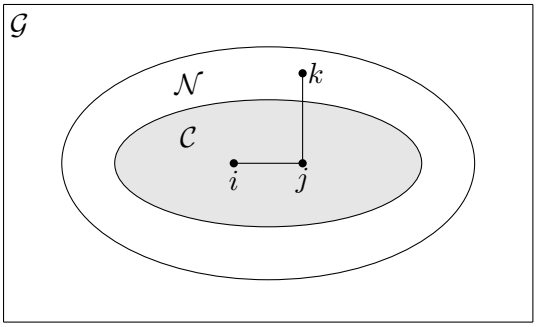
### 3.3.3 Algoritmus Local Optimal Algorithm

Dalším z vybraných algoritmů pro lokální detekci komunit je tzv. Local Optimal Algorithm (dále jen LOA). Definuje strukturu komunity do jisté míry podobně jako detekční algoritmus Clauset. Na rozdíl od Clausetu, LOA nebere v potaz hraniční okolí komunity (boundary). Komunitu označuje jako $\mathcal{C} \subseteq G$, tedy jako podgraf vstupního grafu $G$ [13]. Dále definuje množinu přilehlých hran do komunity, které označuje $\mathcal{N}$. Strukturu komunity dle LOA můžeme vidět na obrázku 10.

Pro posouzení kvality nalezené komunity LOA používá, podobně jako Clauset, lokální modularitu. Výpočet lokální modularity dle LOA můžeme vidět v rovnici 9.

$$M = \frac{\sum_{i,j} S_{ij}\delta(i,j)}{\sum_{i,j} S_{ij}\lambda(i,j)}, \tag{9}$$

kde $S$ nabývá hodnoty 1 pokud vrcholy $i, j$ jsou spojeny hranou, v opačném případě nabývá hodnoty 0. Dále $\delta(i,j)$ nabývá hodnoty 1 pokud se oba dva vrcholy $i, j$ nacházejí v komunitě, v opačném případě nabývá hodnoty 0. A poslední $\lambda(i,j)$ nabývá hodnoty 1 pokud se pouze jeden z vrcholů $i, j$ nachází v komunitě, v opačném případě nabývá hodnoty 0. Jednodušeji řečeno je lokální modularita dle LOA definována jako poměr hran uvnitř komunity a hran přilehlých z neznámého okolí ke komunitě. Na rozdíl od lokální modularity definované u Clausetu, může hodnota lokální modularity být $M \geq 1$. Algoritmus LOA dokonce definuje komunitu jako komponentu/modul, pro nějž platí právě tato podmínka $M \geq 1$ [13].




Obrázek 10: Definice struktury komunity dle LOA

Algoritmus LOA pracuje ve dvou základních fázích. První fáze se nazývá aglomerační. V této fázi algoritmus postupně přidává vrcholy do komunity a propočítává změnu modularity. Pokud se díky dočasnému přidání nového vrcholu do komunity zvýší lokální modularita, vrchol je v komunitě prozatím ponechán. Tento proces funguje iterativně nad kolekcí vrcholů, které jsou sousedy již existujících vrcholů v komunitě. Druhá fáze algoritmu se nazývá optimalizační. V této fázi algoritmus prochází kolekci vrcholů celé komunity a po jednom se vrcholy z komunity odebírají. Pokud odebrání vrcholu způsobí zvýšení lokální modularity a zároveň zůstane komunita jako podgraf souvislý, vrchol zůstane odebrán z komunity. Tím je komunita optimalizovaná. Kontrola souvislosti grafu se dá řešit více způsoby. Pro tuto práci je využit algoritmus prohledávání grafu do šířky (breadth-first search).

Nutno podotknout, že v této fázi se prochází vrcholy komunity mezi než patří i počáteční vrchol (seed). Může nastat situace, že algoritmus odebere v optimalizační fázi i tento vrchol [13]. Dle psedokódu LOA by v takovém případě měl algoritmus skončit se zprávou o nenalezení komunity. Tento scénář není příliš vhodný pro sledování evoluci komunity, což je cílem této práce. Z tohoto důvodu byl algoritmus pro účely této práce modifikován. V případě, že by

v optimalizační fázi algoritmu mělo dojít ke smazání počátečního vrcholu z komunity, algoritmus skončí nalezením komunity obsahující pouze počáteční vrchol (single community).

**Algoritmus 3:** Pseudokód pro algoritmus LOA

```
   input  : Graph G and seed v_0
   output: Community C containing v_0
 1 Inicialization C = ∅, N = ∅;
 2 Add v_0 to C and its neighbors to N;
 3 repeat
 4     create new list Q ;                    /* storage for new adding vertices */
 5     foreach vertex v_j in N do
 6         compute ΔM;
 7         if ΔM > 0 then
 8             add v_j to C;
 9             remove v_j from N;
10             add v_j to Q;
11     repeat
12         create new list deleteQ ;          /* storage for new adding vertices */
13         foreach vertex v_i in C do
14             compute ΔM;
15             if ΔM > 0 and C is connected graph then
16                 remove v_i from C;
17                 add v_i to deleteQ;
18                 add v_i to Q;
19                 if Q contains v_i then
20                     remove v_i from Q;
21     until deleteQ is empty;
22     foreach vertex v_i in Q do
23         foreach neighbor n_j of v_i do
24             if C and N does not contain n_j then
25                 add n_j to N;
26 until Q is empty;
27 if M > 0 then
28     return C;
```

V algoritmu 3 můžeme vidět na řádku 15, v optimalizační fázi, ověřování souvislosti grafu po odstranění vrcholu z komunity. Jak již bylo zmíněno výše, v této práci se souvislost ověřuje pomocí algoritmu prohledávání grafu do šířky. Algoritmus prohledávání do šířky můžeme shrnout

do několika bodů.

**Algoritmus 4:** Pseudokód algoritmu prohledávání grafu do šířky

1 Všem vrcholům nastav příznak signalizující navštívení vrcholu $Visit(vertex) = false$;
2 Počátečnímu vrcholu (seed) nastav $Visit(seed) = true$ a vlož jej do fronty;
3 Z fronty vyjmi první vrchol (x);
4 Pro každý dosud nenavštívený (neobjevený) sousední vrchol $n$ vrcholu $x$ nastav $Visit(n) = true$ a vlož jej na konec fronty;
5 Opakuj kroky 3-4 dokud není fronta prázdná;

Pokud je fronta na konci algoritmu prázdná a každý vrchol grafu má nastavený příznak $Visit(vertex) = true$, graf je souvislý. V opačném případě nikoliv. Asymptotická složitost algoritmu prohledávání grafu do šířky je $\mathcal{O}(|V| + |E|)$, kde $|V|$ je počet vrcholů a $|E|$ je počet hran v prohledávaném grafu. Asymptotická složitost celého algoritmu LOA je $\mathcal{O}(K^2 d)$, kde je $K$ je počet vrcholů komunity a $d$ je průměrný stupeň vrcholů [13].

### 3.3.4 Algoritmus Lancichinetti

Dalším algoritmem pro detekci komunit v komplexních sítích, který byl vybrán pro implementaci, je algoritmus pojmenovaný po jednom z tvůrců Lancichinetti. Tento algoritmus definuje strukturu komunity stejně jako detekční algoritmus LOA v kapitole 3.3.3. Lancichinetti nepracuje s hraničním okolím komunity (boundary). Rozeznává pouze komunitu, která je cílem hledání, a zbývající část prohledávané sítě.

Tento algoritmus hledá komunitu v daném grafu na základě maximalizace hodnoty tzv. $fitness$ funkce

$$f_C = \frac{k_{in}^C}{(k_{in}^C + k_{out}^C)^\alpha}, \tag{10}$$

kde $k_{in}^C$ je počet hran uvnitř komunity a $k_{out}^C$ je počet hran přiléhajících ke komunitě z neznámého okolí [14]. Parametr $\alpha$ ovlivňuje velikost komunity a je uživatelsky nastavitelný. Tato $fitness$ funkce počítá hodnotu, která se vztahuje k celé komunitě. Algoritmus Lancichinetti ještě pracuje s obdobnou variantou této funkce, která svou hodnotu počítá vzhledem ke konkrétnímu vrcholu [14]. Předpis této funkce lze vidět v rovnici 11.

$$f_C^A = f_{C+\{A\}} - f_{C-\{A\}} \tag{11}$$

Výraz $f_{C+\{A\}}$ znamená hodnotu fitness funkce $f_C$ po začlenění vrcholu $A$ do komunity. Analogicky výraz $f_{C-\{A\}}$ znamená hodnotu fitness funkce $f_C$ po odebrání vrcholu $A$ z dosud nalezené komunity. Tento vztah používá Lancichinetti pro posouzení, zda přidat daný vrchol do komunity nebo odebrat.

Algoritmus v každé iteraci prochází množinu všech sousedících vrcholů $\mathcal{N}$ s vrcholy uvnitř komunity $\mathcal{C}$, které samy nepatří do $\mathcal{C}$. V první fázi spočítá hodnotu funkce $f_C^A$ pro všechny vr-

choly v $\mathcal{N}$. Vrchol, který má tuto hodnotu nejvyšší, bude přidán do komunity $C$, čímž vznikne nová větší komunita $C'$. V dalším kroku jsou opět přepočítány hodnoty funkce $f_C^A$ pro všechny vrcholy v nové komunitě $C'$. Pokud bude pro nějaký vrchol hodnota funkce záporná, odebere se z komunity a vznikne nová komunita $C''$. Opět se přepočtou hodnoty $f_C^A$ pro všechny vrcholy v komunitě a takto se proces bude opakovat dále. Jestliže hodnoty funkce pro všechny vrcholy vyjdou kladné, algoritmus pokračuje opět od prvního kroku procházením všech sousedících vrcholů s vrcholy v komunitě. Algoritmus skončí tehdy, když na konci iterace budou mít všechny vrcholy v množině $N$ zápornou hodnotu fitness funkce. Jinými slovy to znamená, že algoritmus dojde do bodu kdy inkluzí jakéhokoliv nového vrcholu z množiny $\mathcal{N}$ do komunity nedojde ke zlepšení kvality komunity [14].

Algoritmus ve své optimalizační fázi může v rámci zlepšení kvality komunity odstranit i samotný počáteční vrchol. V rámci této práce byl tento algoritmus modifikován. Pokud dojde ke smazání počátečního vrcholu z komunity, algoritmus vrátí jako komunitu pouze počáteční

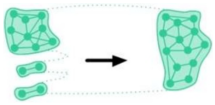
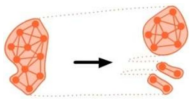
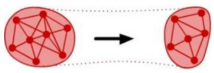
vrchol. Celý postup algoritmu můžeme vidět v algoritmu 5.

**Algoritmus 5:** Pseudokód pro algoritmus Lancichietti

```
   input  : Graph G, seed v_0 and parameter value α
   output: Community C containing v_0
 1 Inicialization C = ∅, N = ∅;
 2 Add v_0 to C;
 3 repeat
 4     N = GetNeighboringVertices();
 5     Add vertex v_max with maximum fitness value to C;
 6     counter=0;
 7     repeat=false;
 8     while true do
 9         if counter==0 or repeat then
10             deleted=0;
11             foreach vertex v_i in C do
12                 if f_C^{v_i} < 0 then
13                     remove v_i from C;
14                     deleted++;
15                     repeat=true;
16                     break;
17             if deleted==0 then
18                 break;
19 until All vertices in N have negative fitness value;
20 if C containing v_0 then
21     return C;
22 else
23     return v_0;
```

# 4 Evoluce lokálních komunit

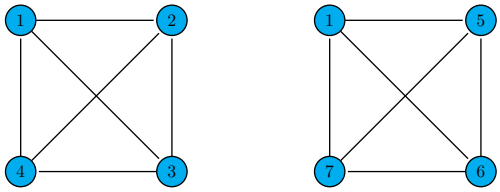
Evoluce komunit popisuje změny v komunitách detekovaných v časových řezech dynamické sítě. Každý časový řez zaznamenává strukturu sítě v daném časovém okamžiku. Pokud chceme sledovat evoluci dané komunity, znamená to, že budeme porovnávat strukturu komunity určené počátečním vrcholem vždy ve dvou po sobě jdoucích časových řezech. Změny struktury komunity v jednotlivých časových okamžicích můžeme popsat událostmi v tabulce 2.

| | | |
|---|---|---|
| **Plynulost** |  | Tato událost říká, že se komunita ve dvou po sobě jdoucích časových řezech nijak zásadně nezměnila. |
| **Zrození** |  | Pokud v časovém řezu $t$ komunita ještě neexistovala a v časovém řezu $t+1$ už ano, nastává událost zrození. |
| **Zánik** |  | Pokud v časovém řezu $t$ komunita již existovala a v časovém řezu $t+1$ už nikoliv, nastává událost zánik. |
| **Spojení** |  | Tato událost říká, že v časovém řezu $t$ existují dvě nebo více komunit, které jsou v časovém řezu $t+1$ spojeny v jednu. |
| **Rozdělení** |  | Událost rozdělení nám říká, že v časovém řezu $t$ existuje komunita, která je v následujícím časovém řezu $t+1$ rozdělena do dvou nebo více. |
| **Nárůst** |  | Nárůst označuje událost, kdy v čase $t$ existuje komunita, která současně existuje i v čase $t+1$ a k tomu se rozrostla o několik vrcholů. |
| **Zmenšení** |  | Zmenšení nám říká, že v čase $t$ existuje komunita, která současně existuje i v čase $t+1$ a její počet vrcholů je menší než v předchozí. |

Tabulka 2: Evoluční události komunit

Existují různé metody detekce těchto událostí mezi dvěma po sobě jdoucími časovými řezy. Některé metody používají množinové operace nad množinami vrcholů a hran obou komunit detekovaných ve dvou po sobě jdoucích časových řezech. Pouhá znalost počtu hran a vrcholů daných komunit nestačí k dostatečně přesné detekci událostí. Jako příklad si můžeme uvést událost plynulosti. Teoreticky může nastat situace taková, kdy komunity mohou mít stejný počet vrcholů i hran, ale samotné vrcholy nebo hrany mezi jednotlivými vrcholy se mohly změnit.

Jinými slovy je třeba znát samotnou identitu vrcholů i hran, abychom mohli dostatečně přesně detekovat jednotlivé události evoluce.




Obrázek 11: Problém s plynulostí dvou komunit detekovaných v časových řezech $t$ a $t+1$

Na podgrafech (komunitách) s počátečním vrcholem $v_0 = 1$ na obrázku 11, můžeme vidět popsaný problém s plynulostí. První komunita je detekována v časovém řezu $t$ a druhá v následujícím časovém řezu $t$+1. Na takto jednoduchých grafech lze vizuálně ověřit, že platí rovnosti $|V_t|=|V_{t+1}|$ a $|E_t|=|E_{t+1}|$. Při bližším prozkoumání zjistíme, že jediný společný vrchol obou komunit je vrchol 1. V časovém řezu $t$+1 se všechny ostatní vrcholy liší, i když je počet vrcholů a struktura grafu zachována. Nejedná se tedy o stejnou komunitu jako v předchozím časovém řezu. Tento problém se objevuje i u ostatních událostí evoluce. V podkapitole 4.2.2 je popsána metoda detekce evolučních událostí, která tento problém řeší.

## 4.1 Metody evolučního vývoje komunit

Algoritmů pro detekci komunit ve statických sítí již bylo v oblasti analýzy síťových dat představeno mnoho. Pojďme se nyní podívat na historický přehled algoritmů, které nějakým způsobem pracují s dynamickými sítěmi a zaznamenávají evoluční události. Jedna z prvních publikací na téma detekce komunit v dynamických sítích byla White a kol. (1976) v [18]. Zkoumali dynamiku sociálních struktur v různých typech sítí. Leskovec a kol. (2005) v [19] se zaměřovali jako jedni z prvních na tendenci růstu u sociálních sítí a na charakteristiky těchto sítí, jako je například distribuce stupňů vrcholů nebo sítě malého světa [17]. Kumar a kol. (2006) v [20] se zaměřovali na charakteristické vlastnosti dvou reálných sítí v různých časových řezech. Popisují pouze charakteristiky na úrovni celé sítě, nikoliv na úrovni komunit v těchto sítích. Falkowski a kol. (2006) v [21] vytvořili metodu pro evoluci komunit. Nad sítěmi v různých časových řezech spustili určitý shlukovací algoritmus. Tento detekční algoritmus je globální, nikoliv lokální [17].

Později se mnozí začali zaměřovat na detekci konkrétních evolučních událostí, které se objevují mezi komunitami detekovanými v různých časových řezech dynamické sítě. Například Palla a kol. (2007) v [22] používají metodu CPM (Clique percolation method) pro detekci evolučních událostí mezi komunitami nalezenými ve dvou po sobě jdoucích časových řezech. Takaffoli a kol. (2011) v [23] nabízejí framework pro detekci evolučních událostí nejen pouze mezi dvě po sobě jdoucími časovými řezy, ale také pro delší časové intervaly [17]. Mucha a kol. (2010) v [24] rozšiřují dynamický přístup založený na maximalizaci modularity k detekci komunit napříč určitým časovým intervalem.

Známou metodou pro evoluci komunit je GED (Group Evolution Discovery). Tato evoluční metoda pracuje výhradně se sociálními sítěmi. Evoluci mezi dvěma komunitami detekuje pomocí inkluze množin hran a vrcholů obou komunit s ohledem na sociální postavení vybraného vrcholu v síti [15].

## 4.2 Metody evoluce vybrané pro implementaci

V následujících dvou podkapitolách jsou objasněny evoluční algoritmy, které byly vybrány k implementaci pro tuto práci.

### 4.2.1 Metoda založena na inkluzi vrcholů

První metoda evoluce, která je implementována, je převzata z [25]. Tato metoda nezohledňuje množiny hran komunit. Pracuje pouze s množinami vrcholů. Tabulka 3 ukazuje množinové inkluze vrcholů komunit pro detekci jednotlivých evolučních událostí mezi časovými řezy $t$ a $t+1$. Pro hodnoty mezního parametru platí podmínka $\nu \in \langle 0; 1 \rangle$.

| **Událost** | **Podmínky pro vrcholy** | | | |
|---|---|---|---|---|
| Plynulost | $V_t \subseteq V_{t+1}$ | $\frac{\lvert V_t \rvert}{\lvert V_{t+1} \rvert} \geq \nu$ | - | - |
| Zrození | $V_t \subseteq V_{t+1}$ | $\frac{\lvert V_{t+1} \cap V_t \rvert}{max(\lvert V_{t+1} \rvert, \lvert V_t \rvert)} \geq \nu$ | - | - |
| Zánik | $V_{t+1} \subseteq V_t$ | $\frac{\lvert V_{t+1} \cap V_t \rvert}{max(\lvert V_{t+1} \rvert, \lvert V_t \rvert)} \geq \nu$ | - | - |
| Nárůst | $V_{t+1} \not\subset V_t$ | $\frac{\lvert V_{t+1} \cap V_t \rvert}{\lvert V_t \rvert} \geq \nu$ | - | - |
| Zmenšení | $V_{t+1} \subseteq V_t$ | $\frac{\lvert V_{t+1} \cap V_t \rvert}{\lvert V_t \rvert} \geq \nu$ | - | - |
| Rozdělení | $\frac{\lvert (V_{t+1} \cup V'_{t+1}) \cap V_t \rvert}{max(\lvert V_{t+1} \cup V'_{t+1} \rvert, V_t)} \geq \nu$ | $\frac{\lvert V_{t+1} \cap V_t \rvert}{\lvert V_{t+1} \rvert} \geq \nu$ | $\frac{\lvert V'_{t+1} \cap V_t \rvert}{\lvert V'_{t+1} \rvert} \geq \nu$ | $\frac{\lvert V'_{t+1} \cap V_t \rvert}{\lvert V_t \rvert} \geq \nu$ |
| Spojení | $\frac{\lvert (V_t \cup V'_t) \cap V_{t+1} \rvert}{max(\lvert V_t \cup V'_t \rvert, V_{t+1})} \geq \nu$ | $\frac{\lvert V_{t+1} \cap V_t \rvert}{\lvert V_t \rvert} \geq \nu$ | $\frac{\lvert V'_t \cap V_{t+1} \rvert}{\lvert V'_t \rvert} \geq \nu$ | $\frac{\lvert V'_t \cap V_{t+1} \rvert}{\lvert V_{t+1} \rvert} \geq \nu$ |

Tabulka 3: Tabulka událostí evoluce v časových řezech $t$ a $t$+1

Jak můžeme vidět v tabulce 3, každá evoluční událost je určena specifickými podmínkami pro množiny vrcholů detekovaných v komunitách v časových řezech $t$ a $t+1$. Tyto podmínky musí platit pro každou evoluční událost v konjunkci. Evoluční metody inkluze vrcholů pro svou práci definuje abstraktní konstrukt s názvem praporek ($community flag$) [25]. Jde o abstraktní identitu komunity, která se mezi jednotlivými časovými řezy může přenášet dál nebo také vytrácet. Jinými slovy se jedná o společný zájem, které mají jednotlivá individua tvořící komunitu. Každá komunita má svou unikátní identitu. Pokud například nastane událost zrození, vznikla nová komunita a s ní vznikla i její unikátní identita. Pokud nastane mezi dvěma časovými řezy událost plynulost, znamená to že identita komunity stále přetrvává a je stabilní [25]. Identita

komunity je definována jako $k\%$ vrcholů komunity, kde hodnota $k$ v tomto kontextu odpovídá hodnotě $\nu \cdot 100$ z tabulky 3 pro jednotlivé evoluční události.

Tato evoluční metoda také popisuje evoluční události pro jednotlivé vrcholy komunit, ale tento přístup nebyl v práci implementován.

### 4.2.2 Metoda založena na inkluzi vrcholů a hran

Metoda založená na množinových inkluzích dvou komunit je další z metod detekce evolučních událostí popsaných v tabulce 2, které mohou nastat ve dvou po sobě jdoucích časových řezech. Předchozí evoluční metoda inkluze vrcholů nebere v úvahu množiny hran komunit. To je vcelku omezující podmínka, protože taková metoda nedokáže do hloubky porovnat rozdíly mezi dvěma komunitami, které mají totožné množiny vrcholů. Takové komunity jsou automaticky plynulé bez ohledu na fakt, že množiny hran mohou být odlišné. Nově navržená metoda zohledňuje i množiny hran v komunitách.

Inkluze počítá procentuální zahrnutí jedné komunity v komunitě druhé. Metoda řeší problém s detekcí událostí, který byl popsán v kapitole 4 na obrázku 11. Podmínky pro vrcholy a hrany můžeme vidět v tabulce 4. V nerovnicích se vyskytují parametry $\nu$ a $\epsilon$, pro které platí $\nu, \epsilon \in \langle 0; 1 \rangle$. Jedná se o rozhodující mezní parametry, které určují hranice pro detekci dané evoluční události. Pro úspěšnou detekci daných evolučních událostí musí být splněny podmínky pro vrcholy a zároveň i pro hrany.

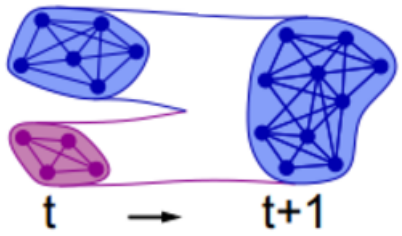
Hodnoty těchto mezních parametrů vyjadřují podobnost mezi oběma komunitami v kontextu dané evoluční události. Pokud například u evoluční události plynulosti zvolíme hodnoty parametrů $\nu = 0.8$ a $\epsilon = 0.8$, znamená to, že množiny vrcholů a hran z časových řezů $t$ a $t + 1$ musí mít společných 80% prvků.

| **Událost** | **Vrcholy** | **Hrany** |
|---|---|---|
| Plynulost | $\frac{\lvert V_t \cap V_{t+1} \rvert}{max(\lvert V_t \rvert, \lvert V_{t+1} \rvert)} \geq \nu$ | $\frac{\lvert E_t \cap E_{t+1} \rvert}{max(\lvert E_t \rvert, \lvert E_{t+1} \rvert)} \geq \epsilon$ |
| Zrození | $V_t \subseteq V_{t+1}$ | $\frac{\lvert E_{t+1} - E_t \rvert}{\lvert E_{t+1} \rvert} \geq \epsilon$ |
| Zánik | $V_t \supseteq V_{t+1}$ | $\frac{\lvert E_t - E_{t+1} \rvert}{\lvert E_t \rvert} \geq \epsilon$ |
| Nárůst | $\frac{\lvert V_{t+1} \cap V_t \rvert}{\lvert V_{t+1} \rvert} \in \langle \nu_1; \nu_2 \rangle$ | $\frac{\lvert E_{t+1} \cap E_t \rvert}{\lvert E_{t+1} \rvert} \geq \epsilon$ |
| Zmenšení | $\frac{\lvert V_t \cap V_{t+1} \rvert}{\lvert V_t \rvert} \in \langle \nu_1; \nu_2 \rangle$ | $\frac{\lvert E_{t+1} \cap E_t \rvert}{\lvert E_t \rvert} \geq \epsilon$ |
| Spojení | $V_t \cup V'_t = V_{t+1}$ | $E_t \cup E'_t \subseteq E_{t+1}$ |
| Rozdělení | $V_{t+1} \cup V'_{t+1} = V_t$ | $E_{t+1} \cup E'_{t+1} \subseteq E_t$ |

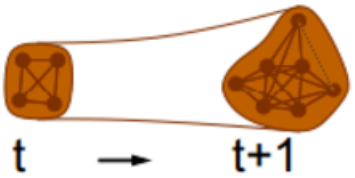
Tabulka 4: Tabulka událostí evoluce v časových řezech $t$ a $t$+1

Události spojení a rozdělení nebudeme ověřovat klasickým výpočtem, který ověřuje procentuální zahrnutí jedné komunity v druhé jako u zbývajících událostí. Nejprve je třeba porozumět rozdílu mezi událostmi nárůstu a spojení. Spojení můžeme vidět detailněji na obrázku 12. V časovém řezu $t$ máme detekovanou komunitu (modrá) $C_t$. V časovém řezu $t$+1 vybraný algoritmus detekoval komunitu $C_{t+1}$, která je nadmnožinou komunity $C_t$. Pokud uděláme rozdíl množin vrcholů a hran komunit, zjednodušeně můžeme znázornit jako $C_{t+1}$ - $C_t$, dostaneme množinu vrcholů a hran přidaných v časovém řezu $t$+1 k původní komunitě $C_t$ detekované v časovém řezu $t$. Pokud tvoří množina nově detekovaných vrcholů v časovém řezu $t$+1 oddělenou komunitu $C'_t$ (na obrázku 12 fialová komunita) v předchozím časovém řezu $t$, lze označit tuto událost jako spojení dvou komunit.

Událost nárůst můžeme vidět na obrázku 13. Analogie názvů detekovaných komunit je podobná jako v případě popisu události spojení s tím rozdílem, že v časovém řezu $t$ postrádáme detekovanou separátní komunitu $C'_t$. U událostí zmenšení a rozdělení platí úplně stejný postup,




Obrázek 12: Spojení komunity $C_t$ a $C'_t$ do $C_{t+1}$. Zdroj: *Community detection in graphs* [16].




Obrázek 13: Nárůst komunity. Zdroj: *Community detection in graphs* [16].

pouze v opačném směru.

# 5 Implementace

Tato kapitola popisuje implementaci aplikace. Aplikace byla napsána na platformě .NET v jazyce C# jako formulářová Winforms aplikace. Pro vývoj bylo použito vývojové prostředí Microsoft Visual Studio 2015. Pro lepší představu o struktuře použitých dat vzhledem k implementaci začneme s popisem datových kolekcí.

## 5.1 Data

Co se týče použitých dat k této práci, byly použity tři typy sítí. První síť byla původně statická (Zachary karate klub). Tato síť byla modifikována do šesti časových řezů. Tato syntetická síť sloužila pro ověření správnosti detekovaných evolučních událostí v počátcích implementace. Vizualizace jednotlivých událostí detekovaných v této syntetické síti je popsána v kapitole 5.3.2.

Dále byly použity syntetické dynamické sítě, které jsou výsledkem optimalizačního SOMA (Self-Organizing Migrating Algorithm) algoritmu. Tento algoritmus byl spuštěn nad různými testovacími funkcemi. Výsledek spuštěného algoritmu nad každou funkcí se rovná jedné množině časových řezů dynamické sítě. Jednotlivé časové řezy jsou v textovém formátu a mají statický počet vrcholů. Pro další popis experimentů nad touto syntetickou dynamickou síti v této práci bude používáno označení SOMA50, SOMA100 a SOMA250, kde číselné označení reprezentuje počet vrcholů v grafu.

Další dynamickou sítí, nad kterou byly prováděny experimenty, je část spoluautorské sítě DBLP. Data byla původně převzata ve formátu SQL (Structured Query Language), kde byly k dispozici časové řezy od roku 2010 až 2015. Každý řez byl reprezentován jako jedna tabulka v databázi. Pro možnost experimentování nad touto sítí byla napsána jednoduchá aplikace, která slouží k načtení řezů této sítě z SQL databáze a k následnému uložení do samostatných textových souborů. Syntaxe textového formátu byla upravena pro nahrání do aplikace pro detekci a evoluci komunit. Strukturu SQL tabulky, která reprezentuje jeden časový řez DBLP, můžeme vidět v tabulce 5.

| **Název sloupce** | **Primární klíč** | **Datový typ** |
|---|---|---|
| Id | Ano | uniqueidentifier |
| AuthorId1 | Ne | uniqueidentifier |
| AuthorId2 | Ne | uniqueidentifier |
| Weight | Ne | int |
| Year | Ne | int |

Tabulka 5: Struktura SQL tabulky reprezentující libovolný časový řez DBLP

Při zpracování dat v tomto formátu byly nejdůležitější sloupce $AuthorId1$, $AuthorId2$ a $Weight$. Sloupce $AuthorId1$ a $AuthorId2$ reprezentují interní identifikátory dvou autorů, kteří společně publikovali článek. Sloupec $Weight$ reprezentuje údaj, který uvádí počet společných

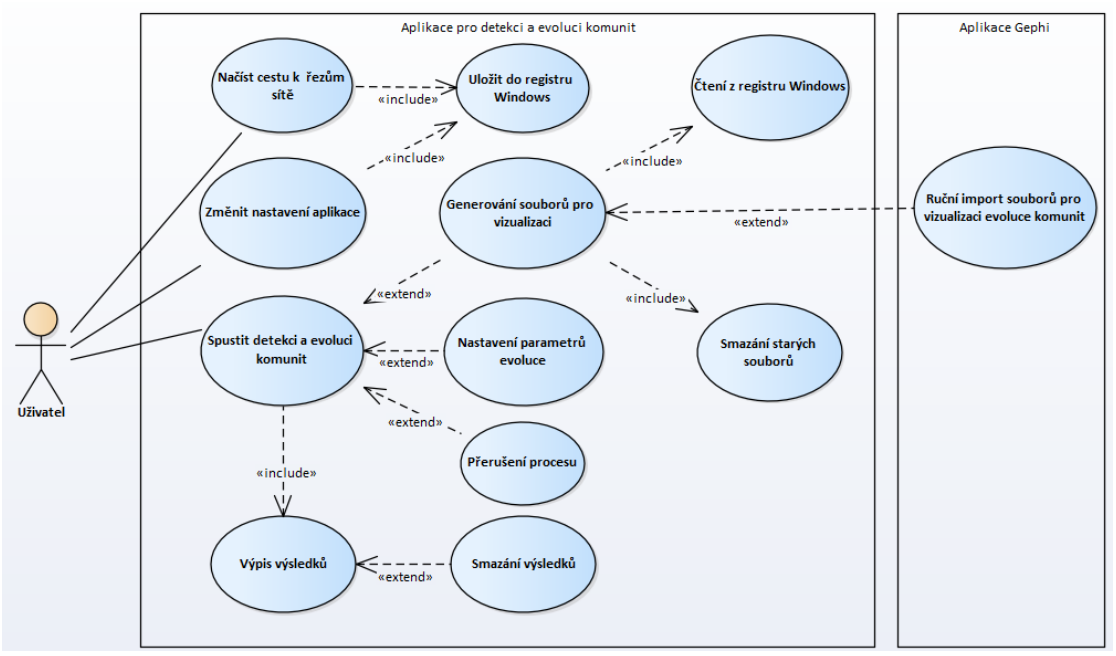
publikací dvou autorů v jednom roce. Sloupec $Id$ reprezentuje pouze primární klíč tabulky. Z toho důvodu se s ním v této práci nepracuje. Sloupec $Year$ reprezentuje rok publikace, se kterým se v této práci také nijak nepracuje. Při prvním pohledu na strukturu tabulky lze vypozorovat, že data jsou anonymizována. Nelze tedy dohledat konkrétní informace o autorech ani publikacích. V tabulce 6 můžeme vidět počty vrcholů a hran jednotlivých sítí v časových řezech 2010 až 2015.

| Časový řez | Počet vrcholů | Počet hran |
|---|---|---|
| 2010 | 182902 | 411260 |
| 2011 | 196572 | 452281 |
| 2012 | 209267 | 492168 |
| 2013 | 218735 | 532387 |
| 2014 | 222788 | 554539 |
| 2015 | 148175 | 353388 |

Tabulka 6: Mohutnosti množin vrcholů a hran pro jednotlivé časové řezy DBLP

## 5.2 Analýza aplikace

Z hlediska analýzy se můžeme zaměřit na statický pohled na aplikaci. V tom případě bude užitečný diagram případů užití na obrázku 14, který ukazuje hranice aplikace pro detekci a evoluci komunit a její uživatelské možnosti.

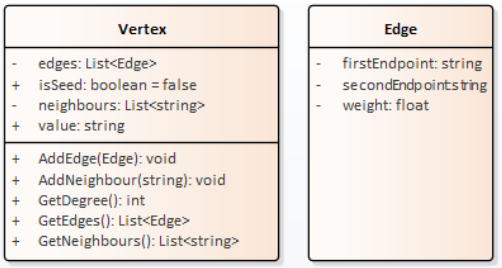



Obrázek 14: Diagram případů užití

Uživateli se nabízí několik možných scénářů jak pracovat s aplikací pro detekci a evoluci komunit. V první řadě má možnost načíst časové řezy dynamické sítě ve vhodném textovém formátu. Dále má uživatel možnost se přes menu dostat do nastavení aplikace, kde může měnit hodnoty parametrů dle potřeby. Detailnější popis nastavení aplikace naleznete v podkapitole 5.3.3. Oba scénáře zahrnují zápis hodnot do registru operačního systému Windows. A v neposlední řadě má uživatel možnost spustit samotný proces detekce a evoluce komunit, který je hlavní funkcí této aplikace. Tento scénář je rozšířen o nepovinný případ užití vizualizace evolučního vývoje komunit. Vizualizace zahrnuje smazání starých souborů pro vizualizaci a je rozšířen o scénář ručního nahrání souborů pro vizualizaci do aplikace Gephi. Proces detekce a evoluce komunit zahrnuje vypsání výsledků případně jejich smazání ze zobrazovacího formulářového prvku.

### 5.2.1 Objektový model a výběr vhodné datové reprezentace

Aplikace pracuje s dvěma třídami, které můžeme vidět na obrázku 15. Nejpoužívanější je třída s názvem *Vertex*, která reprezentuje vrchol grafu. Každá instance této třídy si udržuje svou hodnotu (*value*), což je název vrcholu načtený z řezu sítě. Dále obsahuje vnitřní generickou kolekci svých sousedních vrcholů v daném časovém řezu $t_i$. V případě vážené sítě obsahuje objekt další vnitřní kolekci, která si ukládá hrany (instance třídy *Edge*) i s ohodnocením. Poslední třídou je třída *Edge*, která reprezentuje hranu grafu. V závislosti na typu sítě si ukládá ohodnocení dané hrany nebo nikoliv.




Obrázek 15: Objektový model

Při načítání konkrétního časového řezu dynamické sítě dochází k vytváření objektů typu Vertex a k jejich následnému uložení do příslušné datové kolekce se kterou se později pracuje v procesu detekce komunit. Dále se kontroluje zda, již objekt v kolekci existuje a také se ukládají pro daný vrchol sousední vrcholy, případně hrany. Z počátku vývoje aplikace, kdy se experimentovalo pouze s menšími sítěmi, byla používána generická kolekce typu *List*. V pozdější fázi experimentování nad mnohonásobně větší sítí DBLP se ukázala tato kolekce jako vysoce neefektivní. Kvůli časté kontrole, zda již existuje konkrétní vrchol v kolekci, trvala tato operace u větších řezů (řádově stovky tisíc hran) velice dlouho, protože generický *List* není indexovaná kolekce. Kvůli sekvenčnímu průchodu kolekcí při každém získaní prvku je asymptotická složitost této operace $\mathcal{O}(n)$. Z toho důvodu se vrcholy ukládají do indexované kolekce typu klíč-hodnota

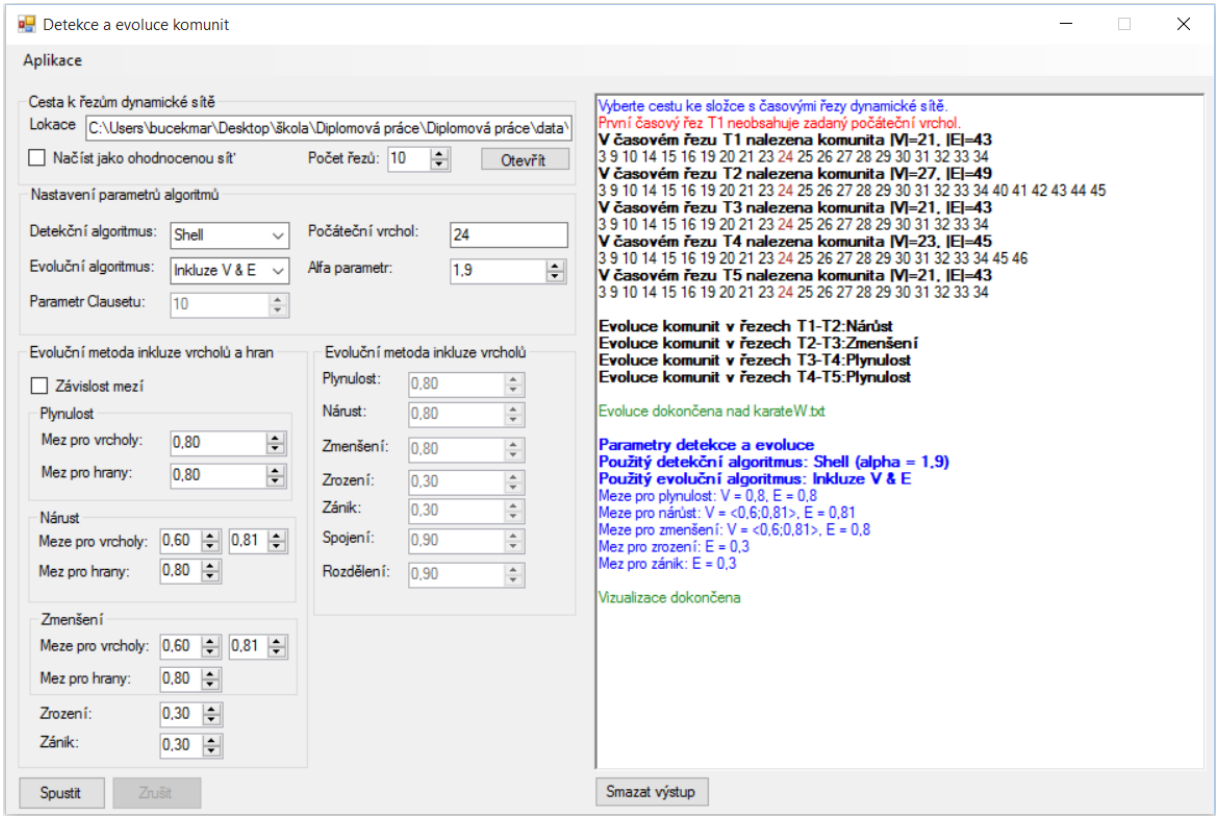
(*Dictionary*), kde jako klíč figuruje atribut *value* objektu $Vertex$ a hodnota je celý objekt $Vertex$. Tento typ kolekce je pro tuto práci velice efektivní, protože získání prvku z kolekce nebo ověření zda prvek v kolekci existuje má asymptotickou složitost pouze $\mathcal{O}(1)$.

## 5.3 Popis aplikace pro detekci a evoluci komunit

Tato kapitola popisuje uživatelské rozhraní aplikace a konkrétní funkce spojené s detekcí a evolucí komunit.

### 5.3.1 Grafické uživatelské rozhraní

Grafické uživatelské rozhraní můžeme vidět na obrázku 16. Vesměs můžeme toto rozhraní rozdělit na ovládací část na levé straně formuláře a zobrazovací část na pravé straně. V horní




Obrázek 16: Grafické rozhraní aplikace

polovině ovládací části formuláře se nachází sekce pro určení lokace složky, kde se nachází časové řezy dynamické sítě v textovém formátu pro načtení do aplikace. V případě, že uživatel chce pracovat i s ohodnocením sítě, musí před nahráním souborů zatrhnout checkbox s názvem Načíst jako ohodnocenou síť.

V další části níže se nachází sekce, která obsahuje formulářové prvky týkající se výběru detekčních a evolučních algoritmů, dále parametry detekčních algoritmů a možnost nastavit

počet časových řezů dynamické sítě, se kterými bude aplikace pracovat. Dále se v této části formuláře nachází prvek pro výběr počátečního vrcholu pro detekci komunit. Co se týče detekce jednotlivých evolučních událostí, uživatel má možnost si přizpůsobit hodnoty mezních parametrů pro každou z těchto událostí. Jedná se o parametry $\nu$ a $\epsilon$, které jsou detailněji popsány v kapitole 4.2.2.

Co se týče zobrazovací části aplikace, nachází se ve formě textboxu v pravé části formuláře. Na obrázku 16 lze vidět, jak vypadá textový výstup po dokončení procesu detekce a evoluce komunit. Po dokončení procesu se zobrazí informace o všech komunitách detekovaných v časových řezech $t_1$ až $t_n$. Mezi tyto informace patří výpis mohutnosti množin hran a vrcholů a všechny vrcholy komunity s barevně odlišeným počátečním vrcholem. Dále se vypíšou detekované evoluční události mezi dvěma po sobě detekovanými komunitami. Pokud má uživatel v nastavení povoleno vypisování parametrů detekce a evoluce, v textboxu se zobrazí po dokončení procesu modrou barvou. Výstupní text má uživatel možnost kdykoliv smazat přes tlačítko pod textboxem.

V závislosti na počtu časových řezů a mohutnosti množin vrcholů a hran jednotlivých komunit, výpočet může trvat delší dobu. Pro tento případ lze přerušit právě probíhající vlákno a pokračovat dále bez nutnosti vypnutí instance aplikace.

### 5.3.2 Vizualizace evoluce komunit

Aplikace používá pro vizualizaci aplikaci Gephi. Po detekci komunit v časových řezech dynamické sítě a následné detekci evolučních událostí mezi jednotlivými komunitami se vygenerují soubory pro Gephi. Generované soubory jsou ve formátu GEXF. Každý soubor v tomto formátu zaznamenává evoluční vývoj komunity v libovolném časovém řezu $t_{i+1}$ oproti předešlému časovému řezu $t_i$.

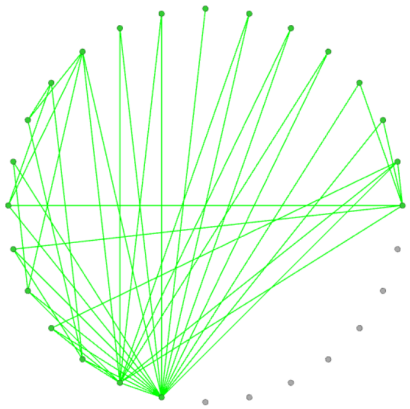
Abychom mohli pozorovat vývoj komunity v čase, je třeba dodržet určitá kritéria pro vizualizaci. V každém řezu se nachází totožná množina vrcholů, která vznikla sjednocením množin vrcholů všech komunit nalezených v časových řezech $t_1$ až $t_n$. Dále je důležité rozmístění vrcholů, které musí být v každém souboru konstantní. V opačném případě by uživatel nemohl pozorovat evoluční události mezi časovými řezy. Z toho důvodu jsou vrcholy rozmístěny staticky do kružnice ve všech řezech.

Zatímco statickou část vizualizace tvoří vrcholy, dynamickou složku tvoří hrany, jejichž počet se v každém časovém řezu $t_i$ mění na základě evolučního vývoje komunity. Pokud se zaměříme na libovolný časový řez $t_i$, jeho množina hran je sjednocením všech množin hran z komunit detekovaných v časových řezech $t_1$ až $t_n$. Jde tedy o všechny hrany, které participovaly v jakémkoliv z těchto časových řezů. Hrany i vrcholy jsou v každém časovém řezu $t_i$ barevně odlišeny na základě evolučních událostí, které nastaly oproti předešlému časovému řezu $t_{i-1}$. Barevné odlišení se tedy týká dvou po sobě jdoucích časových řezů. Význam použitých barev můžeme vidět v tabulce 7.

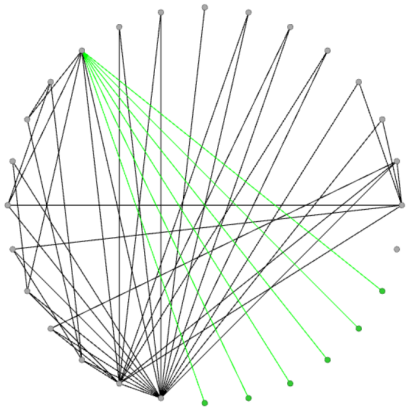
| **Barva** | **Vrcholy\|Hrany** |
|---|---|
| Černá | Hrana se nachází se současně v časovém řezu $t_{i-1}$ a $t_i$. |
| Zelená | V časovém řezu $t_i$ přibyl vrchol nebo hrana |
| Červená | V časovém řezu $t_i$ ubyl vrchol nebo hrana |

Tabulka 7: Tabulka barev použitých při značení evolučních událostí v časových řezech $t_{n-1}$ a $t_n$

Pro názornou ilustraci problému byly uměle vytvořeny 4 časové řezy, jejichž vizualizace můžeme vidět na obrázcích níže. Jde o komunity, které byly detekovány v synteticky vytvořených časových řezech ke Zachary karate klubu. Pro detekci komunit byl použit algoritmus Shell s hodnotou parametru $\alpha = 1.9$ s počátečním vrcholem 24. Na obrázcích 17 a 18 můžeme vidět vývoj



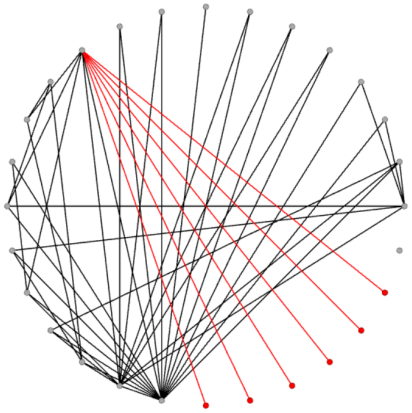
Obrázek 17: Vývoj komunity v časovém řezu $t_1$.



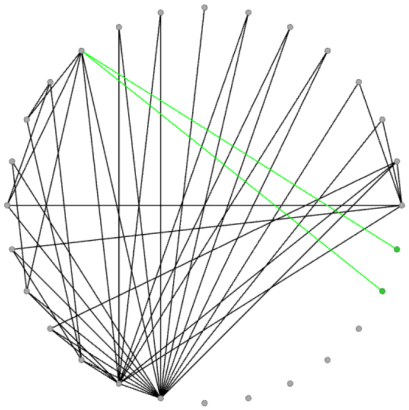
Obrázek 18: Vývoj komunity v časovém řezu $t_2$.

komunit ve dvou po sobě jdoucích časových řezech $t_1$ a $t_2$. Vizualizace vývoje komunity v prvním časovém řezu $t_1$ je jednodušší, protože ji nelze porovnat s žádným předchozím řezem. Dle tabulky 7 můžeme pozorovat barevné odlišení vrcholů a hran. Jelikož celá komunita nyní vznikla, hrany a vrcholy které do této komunity patří jsou obarveny zeleně. Zbývající šedé vrcholy zatím do komunity nepatří. V jednom nebo více z následujících časových řezů budou participovat. V časovém řezu $t_2$ můžeme vidět, že do komunity přibylo šest vrcholů a hran. Vrcholy, které se vyskytovaly v předchozím řezu $t_1$ změnily barvu ze zelené na šedou, protože se vyskytují v obou časových řezech současně. Mezi časovými řezy $t_1$ a $t_2$ nastává evoluční událost nárůst. To samé platí pro hrany obarvené černou barvou. Jeden vrchol zůstává mimo komunitu, taktéž obarven šedou barvou.

Na obrázcích 19 a 20 můžeme vidět vývoj komunit v posledních dvou po sobě jdoucích časových řezech $t_3$ a $t_4$. V časovém řezu $t_3$ ubylo stejných šest vrcholů a hran, které přibyly v předchozím časovém řezu $t_2$. Jejich barva je z toho důvodu červená. Jeden vrchol zůstává stále nezačleněn do evolučního cyklu komunity. Mezi časovými řezy $t_2$ a $t_3$ nastává evoluční událost zmenšení. Do komunity v posledním řezu $t_4$ přibyly dva vrcholy a hrany. Z toho důvodu jsou obarveny zeleně. I přes dva nově přidané vrcholy a hrany nastává evoluční událost plynulost.



Obrázek 19: Vývoj komunity v časovém řezu $t_3$.



Obrázek 20: Vývoj komunity v časovém řezu $t_4$.

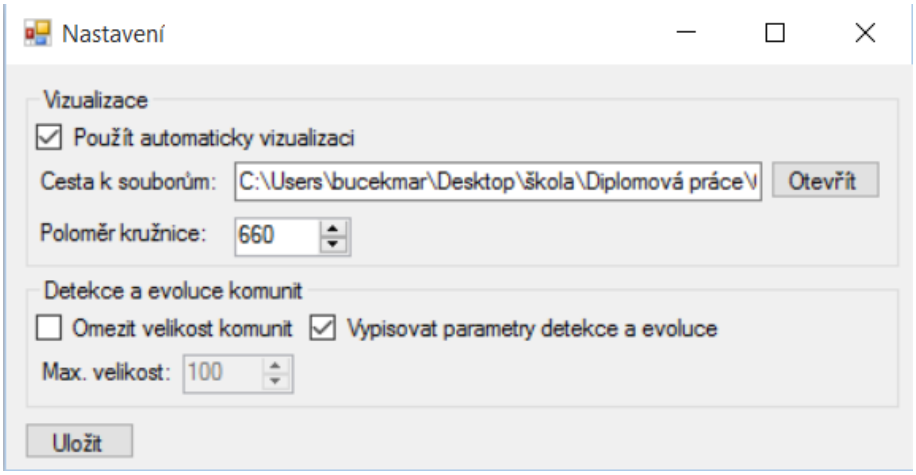
To je způsobeno hodnotou mezního parametru ($\nu = 0,8$) pro plynulost, která byla použita při experimentu.

Může nastat situace, že v některém z časových řezů $t_i$ neparticipuje počáteční vrchol z prvního řezu. Pro tyto případy byly algoritmy pro detekci komunity upraveny tak, aby jejich výstupem byl pouze tento jeden vrchol bez hran. Jde tedy tzv. *singleton* komunitu. V takovém případě vizualizace vykreslí opět množinu vrcholů spolu s červenými hranami, které ubyly.

### 5.3.3 Uživatelské nastavení

Uživatel má možnost si přes kontextové menu v levém horním rohu formuláře přizpůsobit různé nastavení. Formulář tohoto nastavení můžeme vidět na obrázku 21. První z možných nastavení se týká generování souborů pro import a následnou vizualizaci v programu Gephi. Jak již bylo v předešlé kapitole 5.3.2 řečeno, rozmístění vrcholů je koncipováno do kruhového tvaru. Ve vzorci pro výpočet koordinátu v kartézském systému souřadnic se počítá s poloměrem kružnice. Tento poloměr má implicitně nastavenou hodnotu na 650. Pokud jsou množiny vrcholů a hran nalezené komunity příliš velké, výchozí hodnota poloměru kružnice nemusí stačit. V takovém případě by vrcholy sice zachovaly své kruhové rozmístění, ale z důvodu malých vzdáleností mezi vrcholy je taková vizualizace nečitelná. Proto může uživatel nastavit poloměr kružnice dle libosti. Pokud bude uživatel pracovat s menšími sítěmi, může si nastavit poloměr i na menší velikost než je implicitní. Pokud uživatel z nějakého důvodu nebude chtít, aby se automaticky po každé detekci komunit generovaly soubory pro Gephi, může tuto volbu zakázat přes checkbox. Uživatel má také možnost nastavit cestu k ukládání souborů pro vizualizaci. V dané lokaci se po dokončení procesu detekce a evoluce vytvoří složka s názvem gephi, kde se soubory vygenerují.

Další z nastavení je možnost zakázat nebo povolit vypisování parametrů evoluce a detekce. Tyto výpisy jsou užitečné v případě, kdy uživatel experimentuje s různými hodnotami evolučních parametrů. Při zpětném pohledu může uživatel jednoduše dohledat, jaké byly použity hodnoty




Obrázek 21: Uživatelské nastavení aplikace

parametrů u jednotlivých experimentů. Z výpisu také zjistí, jaký detekční algoritmus použil i s jakými hodnotami vstupních parametrů.

Poslední z nastavení je možnost omezit maximální velikost komunit. Při experimentech nad většími sítěmi se ukázalo, že některé detekční algoritmy mají tendenci nalézat komunity, které mají řádově tisíce vrcholů. Taková detekce trvá velice dlouho a zároveň se výsledná komunita nedá čitelně vizualizovat. Z toho důvodu má uživatel možnost použít toto omezení.

Všechny hodnoty v tomto formuláři nastavení se po kliku na tlačítko uloží do registru operačního systému Windows. Při opětovném spuštění aplikace se tyto hodnoty automaticky z registru načtou pro další práci a uživatel je nemusí znova zbytečně nastavovat.

### 5.3.4 Načítání souborů a validace vstupů

Aplikace pro detekci a evoluci komunit pracuje s řezy v textovém formátu a CSV formátu. Pro CSV formát jsou povolené oddělovače mezera a středník. Pokud uživatel použije nepovolený formát nebo pokud nastane jakákoliv chyba při načítání souboru, aplikace o této skutečnosti uživatele informuje zprávou. V takovém případě není možné pokračovat dále v detekci komunit.

Pro úspěšné spuštění detekce a evoluce komunit je nutné správně vyplnit určité vstupy. V případě nesplnění tohoto požadavku aplikace upozorní, že některý ze vstupů není vybrán bez bližší specifikace. První z podmínek je úspěšné načtení časových řezů dynamické sítě. Při prvním spuštění, kdy v registru není uložená žádná cesta, se po otevření dialogu pro výběr složky ukáže implicitně lokace s aplikací. Dále musí uživatel vybrat algoritmus pro detekci komunit, evoluci komunit a počáteční vrchol. Počáteční vrchol musí uživatel napsat ručně. Pokud nebude zadaný vrchol obsažen v první časovém řezu, aplikace na tuto skutečnost upozorní uživatele varovnou zprávou. Dle typu algoritmu bude uživatel moci experimentovat s hodnotami parametrů pro tyto algoritmy. Po splnění těchto požadavků lze spustit proces detekce a evoluce komunit.

### 5.3.5 Načítání a ukládání hodnot do registru operačního systému Windows

Jak již bylo zmíněno dříve, aplikace používá na mnoha místech zápis a čtení z registru operačního systému Windows. Tento postup byl zvolen pro snadnější a rychlejší práci s uživatelským nastavením aplikace. V tabulce 8 můžeme vidět výpis všech hodnot, které aplikace ukládá do registru. Jedná se o hodnoty formulářových prvků, které byly představeny v sekci uživatelské nastavení 5.3.3.

Výchozí hodnota u klíče *Radius* byla zjištěna experimenty s komunitami větších i menších velikostí při vizualizacích v aplikaci Gephi. Hodnota 650 se ukázala pro takto velké sítě jako ideální.

| **Klíč** | **Výchozí hodnota** | **Popis** |
|---|---|---|
| $LastPath$ | Cesta ke spustitelnému souboru aplikace | Cesta ke složce pro načtení řezů sítě |
| $UseVisualization$ | $True$ | Informace zda uživatel chce vizualizovat |
| $GephiLastPath$ | Cesta ke spustitelnému souboru aplikace | Lokace složky k uložení souborů pro Gephi |
| $Radius$ | 650 | Poloměr kružnice pro rozmístění vrcholů při vizualizaci |
| $ReduceCommunityMaxSize$ | $True$ | Informace zda uživatel chce omezit max. velikost komunit |
| $CommunityMaxSize$ | 100 | Maximální velikost komunity |
| $PrintEvolutionParameters$ | $True$ | Informace zda uživatel chce vypisovat parametry detekce a evoluce |

Tabulka 8: Tabulka klíčů použitých pro uložení do registru operačního systému Windows

Všechny klíče z tabulky 8 s uživatelsky zvolenými hodnotami lze nalézt ručně ve stromové struktuře registru v lokaci $HKEY\_CURRENT\_USER/SOFTWARE/CommunityEvolution$.

# 6 Experimenty

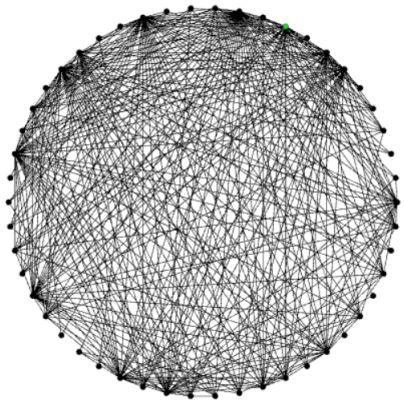
V této kapitole naleznete výsledky experimentů detekce a evoluce komunit nad řezy různých dynamických sítí. Experimenty obsahují porovnání pro jednotlivé detekční i evoluční algoritmy.

## 6.1 Experimenty s lokálními detekčními algoritmy

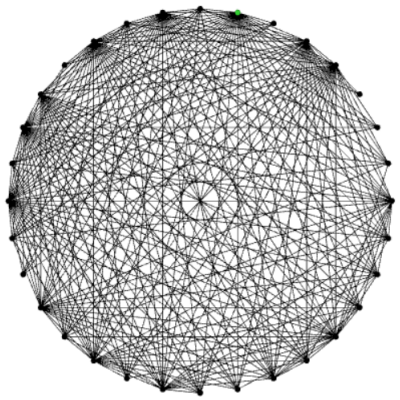
V této podkapitole jsou zhodnoceny výsledky experimentů algoritmů pro detekci komunit v závislosti na jejich vstupních parametrech. Pro vizualizaci detekovaných komunit byla použita podobná metoda jako v sekci 5.3.2. Tato metoda pro experimenty s detekcí komunit vizualizuje statickou strukturu komunity v daném řezu bez evolučních událostí. Počáteční vrcholy jsou odlišeny zelenou barvou.

### 6.1.1 Algoritmus Shell

Prvním testovaným algoritmem pro detekci komunit je Shell. Tento algoritmus lze parametrizovat hodnotou vstupního parametru $\alpha$.



Obrázek 22: Komunita detekovaná v síti SOMA v časovém řezu $t_3$ ($v_0 = 10, \alpha = 2$).

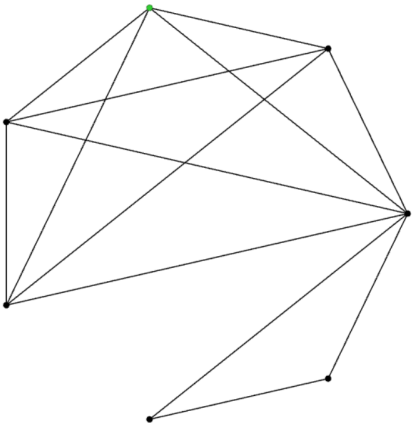


Obrázek 23: Komunita detekovaná v síti SOMA v časovém řezu $t_3$ ($v_0 = 10, \alpha = 3,6$).

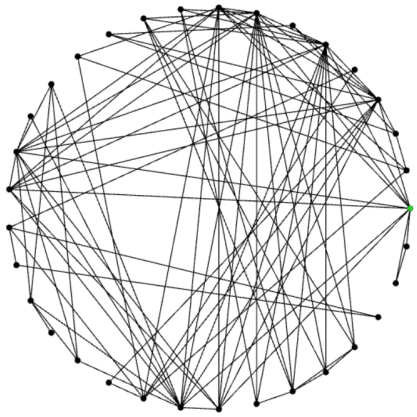
Algoritmus byl spuštěn nad sítí SOMA o velikosti 50 vrcholů pro stejný časový řez s hodnotami parametru $\alpha = 2$ a $\alpha = 3,6$ (obrázek 22 a 23). V prvním případě ($\alpha = 2$) algoritmus obsáhne celou síť. Zvýšením hodnoty parametru $\alpha$ na 3,6 došlo k redukci velikosti detekované komunity z původních 50 vrcholů na 32. Tento algoritmus byl také spuštěn nad DBLP. Pro tuto síť se ukázal jako nevhodný. V mnoha případech byla velikost vrcholů komunity v tisících, což se také projevilo na dlouhém trvání algoritmu. Na základě provedených experimentů lze dojít k závěru, že tento algoritmus detekuje nejpočetnější komunity.

### 6.1.2 Algoritmus LOA

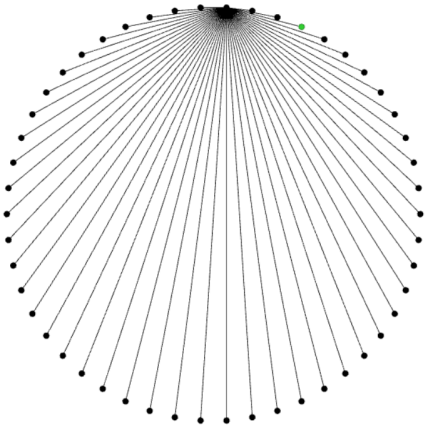
Dalším testovaným detekčním algoritmem je LOA. Tento algoritmus není jako jediný z množiny vybraných detekčních algoritmů pro tuto práci závislý na parametru. Algoritmus byl spuštěn nad sítí DBLP (obrázky 24 a 25) a SOMA50 (obrázky 26 a 27).



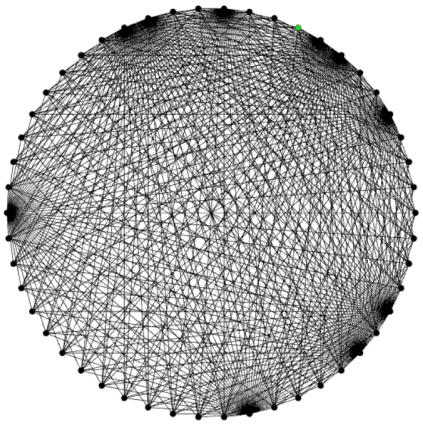
Obrázek 24: Komunita detekovaná v síti DBLP v časovém řezu 2010 ($v_0 = 51$).



Obrázek 25: Komunita detekovaná v síti DBLP v časovém řezu 2011 ($v_0 = 51$).



Obrázek 26: Komunita detekovaná v síti SOMA50 v časovém řezu $t_1$ ($v_0 = 10$).



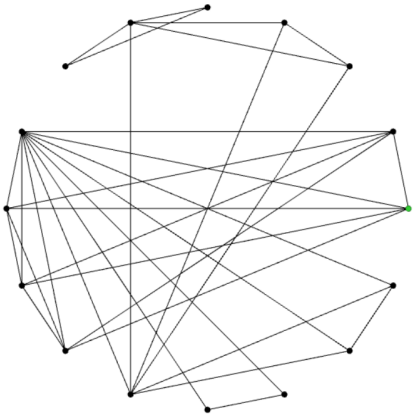
Obrázek 27: Komunita detekovaná v síti SOMA50 v časovém řezu $t_2$ ($v_0 = 10$).

Algoritmus LOA nemá žádný vstupní parametr, kterým by mohl uživatel ovlivňovat velikost komunity. Z toho důvodu byl pro tento experiment zvolen koncept detekce komunity ve dvou po sobě jdoucích časových řezech, ale vždy se stejným počátečním vrcholem.

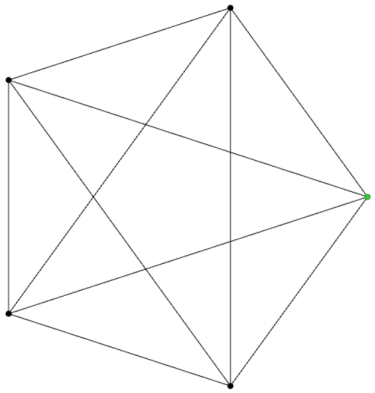
Na obrázku 26 můžeme vidět příklad hubu, tedy vrchol s vysokým stupněm. Koncept hubu byl detailněji probrán v kapitole 2.1.1. Pro dynamickou síť vzniklou výsledkem SOMA algoritmu se ukázal LOA nepříliš efektivní. U všech provedených experimentů nad touto sítí byla výsledná komunita celá původní síť v daném řezu.

### 6.1.3 Algoritmus Lancichinetti

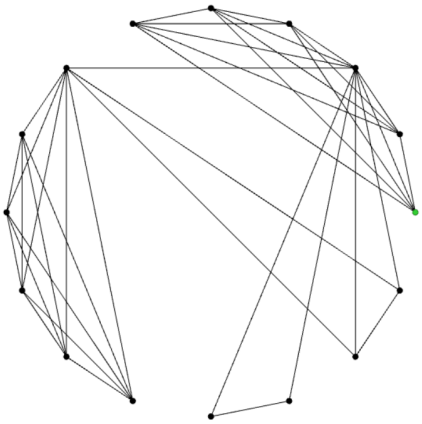
Dalším testovaným detekčním algoritmem je algoritmus Lancichinetti, jehož průběh je závislý na hodnotě parametru $\alpha$. Pro účely této práce byl jako jediný algoritmus upraven pro práci s ohodnocenými sítěmi. Tento algoritmus byl spuštěn nad sítí DBLP v klasické i ohodnocené verzi.



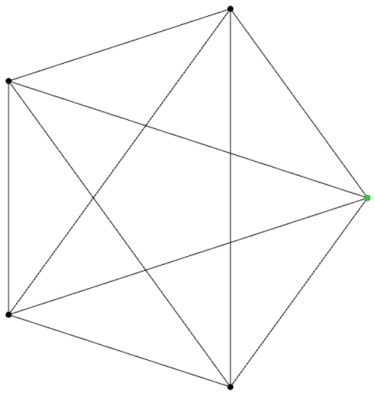
Obrázek 28: Komunita detekovaná v neohodnocené síti DBLP v časovém řezu 2011 ($v_0 = 1, \alpha = 0,5$).



Obrázek 29: Komunita detekovaná v neohodnocené síti DBLP v časovém řezu 2011 ($v_0 = 1, \alpha = 1,9$).



Obrázek 30: Komunita detekovaná v ohodnocené síti DBLP v časovém řezu 2011 ($v_0 = 1, \alpha = 0,5$).



Obrázek 31: Komunita detekovaná v ohodnocené síti DBLP v časovém řezu 2011 ($v_0 = 1, \alpha = 1,9$).

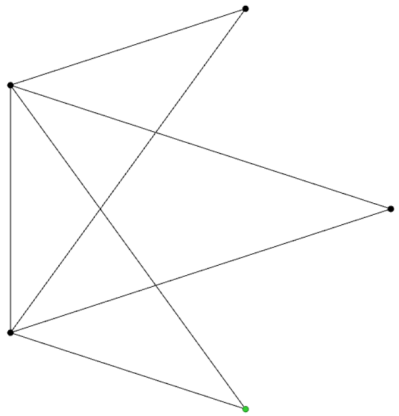
Z porovnání obrázků 28 a 29 u neohodnocené sítě lze vidět, že v případě malé hodnoty parametru $\alpha$ je algoritmus více benevolentnější v přidávání vrcholů do komunity. Výsledkem je mnohonásobně větší komunita. V případě větších hodnot je algoritmus striktnější a výsledkem je analogicky menší komunita. Tento vztah platí úplně stejně i pro komunity detekované v ohodnocené síti (obrázky 30 a 31).

Zároveň se nabízí porovnání komunit detekovaných v neohodnocené a ohodnocené síti se stejným počátečním vrcholem a hodnotou parametru $\alpha$. V obou typech sítí byl algoritmus spuštěn se stejným počátečním vrcholem a stejnou hodnotou parametru $\alpha$. Pokud porovnáme komunity na obrázcích 28 a 30, zjistíme že jde o odlišné komunity. Na první pohled sice mají stejné mohutnosti množin vrcholů, ale jedná se o odlišné vrcholy. V druhém případě (obrázky 29 a 31) našel detekční algoritmus totožné komunity bez ohledu na váženost sítě.

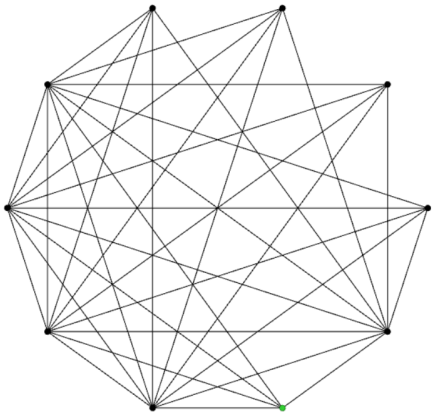
Jak již bylo zmíněno dříve, algoritmus Lancichinetti má vstupní parametr, který ovlivňuje velikost komunity. Z toho důvodu byl algoritmus spuštěn s různými hodnotami parametru $\alpha$ nad stejným časovým řezem i počátečním vrcholem. Tento koncept byl vybrán kvůli ukázky vlivu změny hodnoty parametru. Totéž platí pro experimenty s ohodnocením sítě.

### 6.1.4 Algoritmus Clauset

Dalším testovaným detekčním algoritmem je algoritmus Clauset, jehož průběh je závislý na hodnotě parametru $k$, který udává maximální možnou velikost detekované komunity.

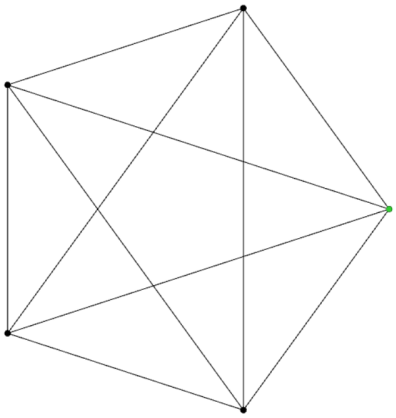


Obrázek 32: Komunita detekovaná v síti SOMA50 v časovém řezu $t_2$ ($v_0 = 15, k = 5$).

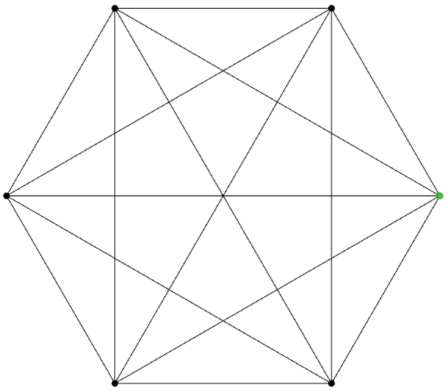


Obrázek 33: Komunita detekovaná v síti SOMA50 v časovém řezu $t_2$ ($v_0 = 15, k = 10$).

Algoritmus byl spuštěn nad sítěmi SOMA50 (obrázky 32 a 33) DBLP s hodnotami parametrů $k = 5$ a $k = 10$ (obrázky 34 a 35). Při porovnání jednotlivých experimentů nad časovými řezy SOMA50 můžeme vidět, že algoritmus vždy expandoval k maximální možné velikosti komunity v závislosti na hodnotě parametru $k$. V prvním případě byl počet vrcholů $|V| = 5$ a v druhém případě $|V| = 10$, což odpovídá nastaveným hodnotám parametru. U experimentů s časovými řezy DBLP se choval algoritmus trochu jinak. Experiment byl spuštěn se stejnými hodnotami parametru $k$ jako u SOMA50, ale u experimentu s $k = 10$ měla nalezená komunita mohutnost množiny vrcholů $|V| = 6$. Algoritmus tedy ne vždy dosáhne předem definované velikosti komunity. Běh algoritmu byl ukončen dříve z důvodu dosažení maximální hodnoty lokální modularity.



Obrázek 34: Komunita detekovaná v síti DBLP v časovém řezu 2011 ($v_0 = 1, k = 5$).



Obrázek 35: Komunita detekovaná v síti DBLP v časovém řezu 2011 ($v_0 = 1, k = 10$).

## 6.2 Experimenty s evolucí komunit

Tato kapitola obsahuje výsledky experimentů s algoritmy pro evoluci lokálních komunit. Pro přehlednější a stručnější zápis jsou jednotlivé evoluční události uváděny zkratkami ve sledu. Jejich výčet můžeme vidět v tabulce 9. Zároveň zde nalezneme hodnoty pro detekci evolučních událostí, které byly použity při experimentech.

| **Událost** | **Zkratka** | **Metoda evoluce 1** | **Metoda evoluce 2** |
|---|---|---|---|
| Plynulost | $P$ | $\nu = 0,8$ | $\nu = 0,8$; $\epsilon = 0,8$ |
| Zrození | $Zr$ | $\nu = 0,3$ | $\epsilon = 0,3$ |
| Zánik | $Z$ | $\nu = 0,3$ | $\epsilon = 0,3$ |
| Spojení | $S$ | $\nu = 0,9$ | - |
| Rozdělení | $R$ | $\nu = 0,9$ | - |
| Nárůst | $N$ | $\nu = 0,8$ | $\nu_d = 0,6$; $\nu_h = 0,81$; $\epsilon = 0,8$ |
| Zmenšení | $Zm$ | $\nu = 0,8$ | $\nu_d = 0,6$; $\nu_h = 0,81$; $\epsilon = 0,8$ |
| Žádná událost | - | - | - |

Tabulka 9: Tabulka zkratek evolučních událostí a hodnot parametrů pro detekci evolučních událostí použitých při experimentech

Evoluční metodou 1 je myšlena evoluční metoda založena na inkluzi vrcholů a evoluční metoda 2 je evoluční metoda založena na inkluzi vrcholů a hran. Indexy parametru $\nu$ (d,h) u evoluční metody inkluze hran a vrcholů značí dolní a horní mez pro vrcholy. Pro experimenty s evolucí komunit byl doimplementován algoritmus Neighbourhood, který vrací sousední vrcholy zadaného počátečního vrcholu $v_0$ včetně $v_0$. Tento algoritmus byl vytvořen z toho důvodu, abychom věděli jak se v jednotlivých časových řezech mění okolí zadaného počátečního vrcholu. Výsledky experimentů pro algoritmus Neighbourhood můžeme vidět spolu s detekčními algoritmy. Pro experimenty bylo použito sekvenční ověřování jednotlivých evolučních událostí.

### 6.2.1 Experimenty nad časovými řezy SOMA50

Jak již bylo zmíněno v kapitole 5.1, v rámci této práce se experimentovalo nad časovými řezy dynamické sítě SOMA algoritmu. Pro tuto práci byly použity sítě různých velikostí. V tabulce 11 můžeme vidět výsledky detekce a evoluce komunit nad časovými řezy sítě, jejíž řezy mají konstantně 50 vrcholů a počty hran se pohybují řádově v desítkách až stovkách. Jednotlivé experimenty byly spuštěny s pro počáteční vrcholy 10, 20 a 30. Pro tyto experimenty byla použita evoluční metoda s inkluzi vrcholů (tabulka 10) a evoluční metoda s inkluzí hran i vrcholů (tabulka 11). Experimenty byly provedeny nad prvními deseti časovými řezy.

| **Detekční algoritmus** | $v_0 = 10$ | $v_0 = 20$ | $v_0 = 30$ |
|---|---|---|---|
| Shell ($\alpha = 1,9$) | NPPPPPPPR | NRZmPPZmPPP | NPPPPPPPP |
| LOA | PPPPPPPPP | PPPPPPPPP | PPPPPPPPP |
| Lancichinetti ($\alpha = 1,9$) | PZZm-PPP-N | PZ-PZmZm-PP | PZPZmPP- -P |
| Clauset ($k = 10$) | - -PNP-NN- | - - -NP-NNP | - -P-P- -N- |
| Neighbourhood | N-PPPPPPP | N-ZmPPZmPPP | N-PPZmPPP |

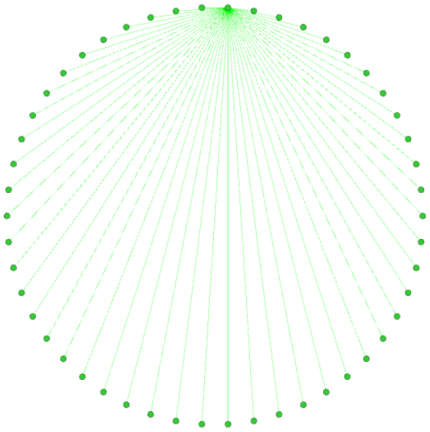
Tabulka 10: Tabulka detekovaných evolučních událostí nad časovými řezy SOMA50 s použitím evoluční metody inkluze vrcholů

| **Detekční algoritmus** | $v_0 = 10$ | $v_0 = 20$ | $v_0 = 30$ |
|---|---|---|---|
| Shell ($\alpha = 1,9$) | ZrZPPPPPPZm | ZrZPPPPPPP | ZrZPPPPPPP |
| LOA | ZrZPPPPPPP | ZrZPPPPPPP | ZrZPPPPPPP |
| Lancichinetti ($\alpha = 1,9$) | ZrZ- -PPP- - | ZrZ-P- - -PP | ZrZP-PP- -P |
| Clauset ($k = 10$) | - - PPP- - - - | - - ZmPP- - -P | - -P-P- - - - |
| Neighbourhood | Zr-PPPPPPP | Zr-PPPPPPP | Zr-PPPPPPP |

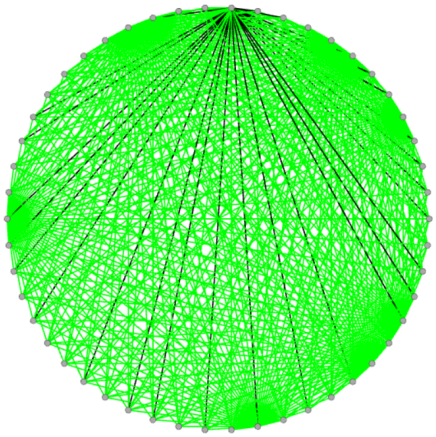
Tabulka 11: Tabulka detekovaných evolučních událostí nad časovými řezy SOMA50 s použitím evoluční metody inkluze hran a vrcholů

Mezi oběma evolučními metodami lze pozorovat v určitých případech veliké rozdíly. Například u algoritmu Lancichinetti můžeme vidět velký rozdíl v první detekované evoluční události. Při použití evoluční metody inkluze hran a vrcholů byla pro všechny počáteční vrcholy detekována první evoluční událost zrození, zatímco u evoluční metody inkluze vrcholů to byla událost plynulost. V obou případech algoritmus Lancichinetti detekoval komunitu v časovém řezu $t_1$ o velikosti |V|=50, |E|=49. V dalším časovém řezu $t_2$ detekoval komunitu o velikosti |V|=50, |E|=445. Z pohledu první evoluční metody jde o zrození, protože přibylo velké procento hran. Z pohledu druhé evoluční metody jde o plynulost, protože mají množiny vrcholů obou komunit jsou stejné a s hranami nijak nepočítá.

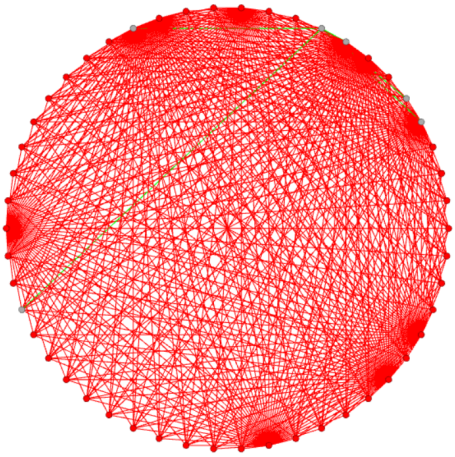
Další rozdíl mezi evolučními algoritmy lze vidět u experimentu s algoritmem Clauset a počátečním vrcholem $v_0$=10. Metoda inkluze vrcholů detekuje dvakrát po sobě evoluční událost



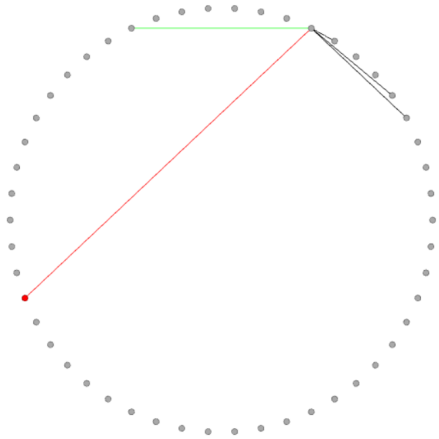
Obrázek 36: Komunita detekovaná v síti SOMA50 v časovém řezu $t_1$ ($v_0 = 10$).



Obrázek 37: Evoluční vývoj komunity mezi časovými řezy $t_1$ a $t_2$.



Obrázek 38: Evoluční vývoj komunity mezi časovými řezy $t_2$ a $t_3$.



Obrázek 39: Evoluční vývoj komunity mezi časovými řezy $t_3$ a $t_4$.

nárůst, zatímco metoda inkluze hran a vrcholů nedetekuje žádnou událost. Z pohledu metody inkluze vrcholů jde o nárůst, protože do množiny vrcholů přibyly nové vrcholy a s hranami už nepracuje. Z pohledu druhé evoluční metody není detekována žádná událost, protože vrcholy sice přibyly, ale také se změnily množiny hran. Jelikož musí platit podmínky pro vrcholy i hrany v konjunkci, událost nárůst v tomto případě není detekována. Pro lepší názornost příkladů můžeme na obrázcích 36, 37, 38 a 39 pozorovat evoluční vývoj komunity detekované algoritmem Lancichinetti v prvních čtyřech časových řezech.

Datová kolekce se ukázala jako nepříliš vhodná pro experimenty. Většina použitých detekčních algoritmů měla tendenci obsáhnout celou síť jako výslednou komunitu.

### 6.2.2 Experimenty nad časovými řezy SOMA50 s odlišnými evolučními parametry

Datová kolekce ukázala být z výsledků předešlých experimentů plynulá. Nyní se můžeme podívat jak budou vypadat výsledky dalších experimentů, když zvýšíme mez pro vrcholy a hrany pro detekci evoluční události plynulosti u obou evolučních algoritmů. Hodnoty mezních parametrů pro detekci plynulosti jsou pro tento experiment $\nu = 0,98$ a $\epsilon = 0,98$. Hodnoty mezních parametrů pro zbývající evoluční události jsou stejné jako v tabulce 9. Výsledky experimentů pro obě evoluční metody můžeme vidět v tabulkách 12 a 13.

| **Detekční algoritmus** | $v_0 = 10$ | $v_0 = 20$ | $v_0 = 30$ |
|---|---|---|---|
| Shell ($\alpha = 1,9$) | NPPPPPPPR | NRZmPPZmPPP | NPPPPPPPP |
| LOA | PPPPPPPPP | PPPPPPPPP | PPPPPPPPP |
| Lancichinetti ($\alpha = 1,9$) | PZZm-PPP-N | PZ-PZmZm-PP | PZPZmPP- -P |
| Clauset ($k = 10$) | - -PNP-NN- | - - -NP-NNP | - -P-P- -N- |
| Neighbourhood | N-PPPPPPN | N-ZmPPZmPPP | N-PNZmPZmPP |

Tabulka 12: Tabulka detekovaných evolučních událostí nad časovými řezy SOMA50 s použitím evoluční metody inkluze vrcholů

| **Detekční algoritmus** | $v_0 = 10$ | $v_0 = 20$ | $v_0 = 30$ |
|---|---|---|---|
| Shell ($\alpha = 1,9$) | ZrZ- -P- - -Zm | ZrZ- -P- - -P | ZrZ- -P- - -P |
| LOA | ZrZ- -P- - -P | ZrZ- -P- - -P | ZrZ- -P- - -P |
| Lancichinetti ($\alpha = 1,9$) | ZrZ- -PPP- - | ZrZSP- - -PP | ZrZP-PP- -P |
| Clauset ($k = 10$) | - -P-P- - - - | - -Zm- - - - -P | - - - -P- - - - |
| Neighbourhood | Zr- - -P- - - - | Zr- - -P- - -P | Zr- - - - - -PP |

Tabulka 13: Tabulka detekovaných evolučních událostí nad časovými řezy SOMA50 s použitím evoluční metody inkluze hran a vrcholů

Při porovnání výsledků tohoto experimentu s předešlým můžeme vidět rozdíl v detekovaných evolučních událostech. Zvýšení hodnot mezních parametrů $\nu$ a $\epsilon$ pro plynulost z původních 0,8 na 0,98 změnilo detekované evoluční události. Největší rozdíly můžeme pozorovat u evoluční metody inkluze vrcholů a hran v tabulce 13. U všech detekčních algoritmů použitých pro tento experiment můžeme vidět, že evoluční událost plynulost byla detekována v menším měřítku než v předešlém experimentu s menšími hodnotami $\nu$ a $\epsilon$. Jelikož byly v tomto experimentu spuštěny detekční algoritmy se stejnými parametry, vizualizace evolučního vývoje komunit je stejná jako v předešlém experimentu.

### 6.2.3 Experimenty nad časovými řezy DBLP

Další datovou kolekcí nad kterou byly provedeny experimenty s detekcí a evolucí komunit je DBLP. Experimenty nad touto datovou kolekcí byly spuštěny opět s hodnotami parametrů, které

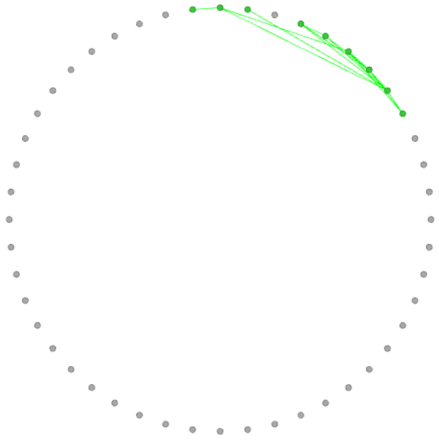
jsou uvedeny v tabulce 9. Datová kolekce obsahuje šest časových řezů. Výsledky experimentů pro obě evoluční metody můžeme vidět v tabulkách 14 a 15. Na obrázcích 40, 41, 42 a 43 můžeme vidět evoluční vývoj komunity detekované algoritmem LOA.

| **Detekční algoritmus** | $v_0 = 147125$ | $v_0 = 1$ |
|---|---|---|
| Shell ($\alpha = 1,9$) | -N- - - | -ZPPP |
| LOA | - -Z- - | -ZPPP |
| Lancichinetti ($\alpha = 0,5$) | - - - - - | -ZPPP |
| Lancichinetti ohodnocený ($\alpha = 0,5$) | - - - - - | -ZPPP |
| Clauset ($k = 10$) | - - - - - | -ZPPP |
| Neighbourhood | -N- - - | -ZPPP |

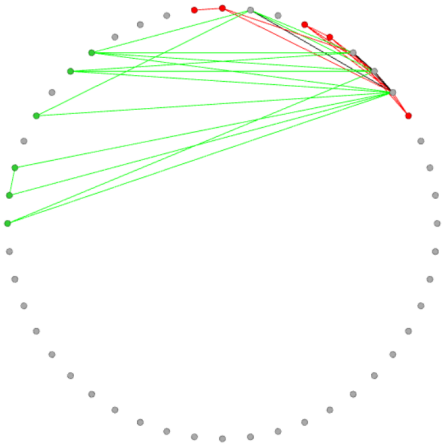
Tabulka 14: Tabulka detekovaných evolučních událostí nad časovými řezy DBLP s použitím evoluční metody inkluze vrcholů

| **Detekční algoritmus** | $v_0 = 147125$ | $v_0 = 1$ |
|---|---|---|
| Shell ($\alpha = 1,9$) | - - - - - | -Z- - - |
| LOA | - -Z- - | -Z- - - |
| Lancichinetti ($\alpha = 0,5$) | -Z- - - | -Z- - - |
| Lancichinetti ohodnocený ($\alpha = 0,5$) | -Z- - - | -Z- - - |
| Clauset ($k = 10$) | - - - - - | -Z- - - |
| Neighbourhood | - -Z- - | -Z- - - |

Tabulka 15: Tabulka detekovaných evolučních událostí nad časovými řezy DBLP s použitím evoluční metody inkluze hran a vrcholů

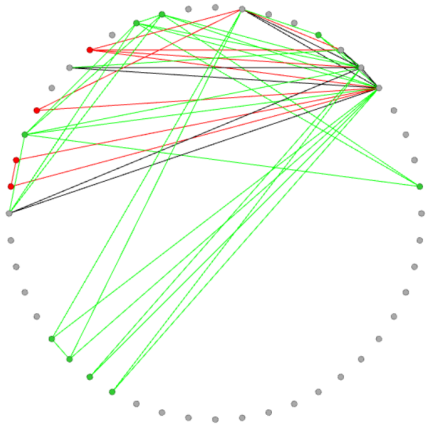


Obrázek 40: Komunita detekovaná v síti DBLP v časovém řezu 2010 ($v_0 = 147125$).

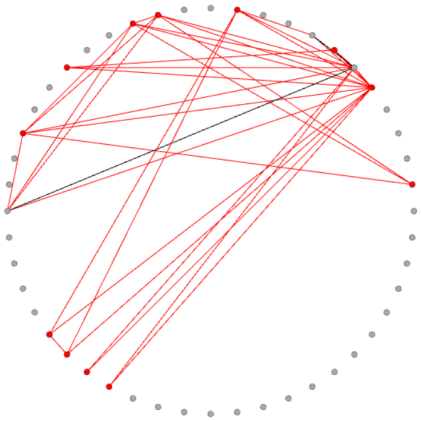


Obrázek 41: Evoluční vývoj komunity mezi časovými řezy 2010 a 2011.

DBLP se ukázala pro experimentování jako z určitého pohledu nevhodná z toho důvodu, že ne všechny časové řezy obsahují vrcholy z prvního časového řezu $t_1$. Pokud uživatel spustí proces detekce komunit s určitým počátečním vrcholem $v_0$, který se nevyskytuje v některých z



Obrázek 42: Evoluční vývoj komunity mezi časovými řezy 2011 a 2012.



Obrázek 43: Evoluční vývoj komunity mezi časovými řezy 2012 a 2013.

následujících časových řezů $t_i$ až $t_n$, výsledkem je prázdná komunita. Z toho důvodu byly detekční algoritmy upraveny, aby v tomto specifickém případě vrátily komunitu obsahující právě jeden vrchol $v_0$. Dalším omezením byla velikost detekovaných komunit. V některých případech byla mohutnost množin vrcholů komunity v řádech tisíců. Z toho důvodu může uživatel omezit velikost komunit. Toto omezení je detailněji probráno v kapitole uživatelského nastavení 5.3.3. Pro tento experiment nad DBLP bylo zavedeno omezení na maximální velikost komunity na hodnotu 20. Z experimentu je patrné, že komunity detekované v časových řezech 2010 až 2015 se strukturálně lišily. Z toho důvodu evoluční algoritmy ve většině případů nedetekovaly žádnou evoluční událost. Tytéž výsledky jsme dostali při zvýšení maximální velikosti komunity na hodnotu 100.

# 7 Závěr

Experimenty s algoritmy pro detekci komunit prokázaly, že struktura síťových dat má zásadní vliv na strukturu detekovaných komunit. Jako příklad lze uvést síť SOMA50. Experimenty s detekčními algoritmy nad touto dynamickou sítí v mnoha případech skončily obsáhnutím celé sítě. Také hodnoty vstupních parametrů některých algoritmů ovlivňují výsledek. Při experimentech s detekcí komunit se ukázalo, že i různé algoritmy mohou detekovat stejné komunity. Jako příklad lze uvést experimenty s algoritmy Lancichinetti a Clauset, které nad sítí DBLP pro $v_0 = 1$ detekovaly stejné komunity. Během experimentování by se měl důkladně zvážit výběr vhodných detekčních algoritmů, případně výběr vhodných hodnot parametrů v závislosti na datech.

Z výsledků experimentů s evolučními algoritmy bylo zjištěno, že hodnoty nastavitelných parametrů implementovaných evolučních algoritmů hrají značnou roli v detekci evolučních událostí. Evoluční algoritmy pracují s komunitami, které byly detekovány vybraným detekčním algoritmem. I samotný výběr detekčního algoritmu hraje roli ve výsledcích evoluce, protože komunity se mohou lišit. U některých experimentů bylo zjištěno, že evoluční události byly u rozdílných detekčních algoritmů podobné. Jako příklad lze uvést experiment nad datovou kolekcí SOMA50. V prvním experimentu můžeme vidět podobnost mezi výsledky evoluce u algoritmů Shell a LOA nebo také Neighbourhood a Shell.

Práce nabízí několik možností rozšíření nebo navázání na toto téma. Množina vybraných algoritmů pro lokální detekci a evoluci komunit může být rozšířena o další algoritmy. Lze modifikovat časové okénko, které v této práci pracuje s dvěma po sobě jdoucími časovými řezy. Práce může být rozšířena o časové okénko, které zachycuje jiný časový úsek.

Tato práce se zabývá implementací algoritmů pro lokální detekci komunit a popisem jejich evolučního vývoje v časových řezech dynamických sítí. Charakteristické vlastnosti komplexních sítí a různé pohledy na definice komunit byly probrány v druhé kapitole. Rozdělení různých přístupů k detekci komunit a vybrané detekční algoritmy jsou probrány v třetí kapitole této práce. V další kapitole byly popsány jednotlivé události evoluce a způsoby jejich detekce v časových řezech spolu s popisem vybraných evolučních algoritmů k implementaci. A v poslední řadě je popsána implementace aplikace a výsledky experimentů nad časovými řezy vybraných dynamických sítí.

# Literatura


[1] Petr Kovář. *Úvod do Teorie grafů.* Vysoká škola báňská – Technická univerzita Ostrava, 2012.

[2] Jongkwang Kim, Thomas Wilhelm. *What is a complex graph?* Theoretical Systems Biology, FLI Jena, Beutenbergstrasse 11, D-07745 Jena, Germany, 2007.

[3] Albert-László Barabási. *NETWORK SCIENCE THE SCALE-FREE PROPERTY.* `http://barabasi.com/f/623.pdf`

[4] Albert-László Barabási, Eric Bonabeau. *Scale-Free Networks.* 2003. `http://barabasi.com/f/124.pdf`

[5] Jeffrey Travers, Stanley Milgram. *An Experimental Study of the Small World Problem.* American Sociological Association. 1969.

[6] Réka Albert, Hawoong Jeong, Albert-László Barabási. *Diameter of the World-Wide Web.* Department of Physics, University of Notre Dame. 1999.

[7] Fortunato Santo. *Community detection in graphs.* Complex Networks and Systems Lagrange Laboratory, ISI Foundation, Viale S. Severo 65, 10133, Torino, I-ITALY., 2010.

[8] Ashish Kumar Patnaik, Prasanta Kumar Bhuyan, K.V. Krishna Rao. *Divisive Analysis (DIANA) of hierarchical clustering and GPS data for level of service criteria of urban streets.* 2015.

[9] Kelbel Jan, Šilhán David. *Shluková analýza.* České vysoké učení technické, Fakulta kybernetiky.

[10] Bagrow James, Erik Bollt. *A Local Method for Detecting Communities.* Department of Physics, Clarkson University, Potsdam, NY 13699-5820, USA, 2008.

[11] Aaron Clauset. *Finding local community structure in networks.* Department of Computer Science, University of New Mexico, Albuquerque NM 87131, 2008.

[12] Shixiong Xia, Ranran Zhou, Yong Zhou, Mu Zhu. *An Improved Local Community Detection Algorithm Using Selection Probability.* School of Computer Science and Technology, China University of Mining and Technology, Xuzhou 221116, China, 2014.

[13] Feng Luo, James Z. Wang, Eric Promislow. *Exploring Local Community Structures in Large Networks.* Department of Computer Science, 100 McAdams Hall, Clemson University, Clemson, SC 29634-0974, USA, 2006.

[14] Andrea Lancichinetti, Santo Fortunato, János Kertész. *Detecting the overlapping and hierarchical community structure in complex networks.* Complex Networks Lagrange Laboratory

(CNLL), Institute for Scientific Interchange (ISI), Viale S. Severo 65, 10133, Torino, Italy, 2009.

[15] Piotr Bródka, Stanisław Saganowski, Przemysław Kazienko. *GED: the method for group evolution discovery in social networks.* Wrocław University of Technology, Wyb.Wyspiańskiego 27, 50-370 Wrocław, Poland, 2012.

[16] Daniel Estrada. *Community detection in graphs.* `http://digitalinterface.blogspot.cz/2013/05/community-detection-in-graphs.html`

[17] Mansoureh Takaffoli, Farzad Sangi, Justin Fagnan, Osmar R. Zaiane. *Community Evolution Mining in Dynamic Social Networks.* Department of Computing Science, University of Alberta, 2011.

[18] Harrison C. White, Scott A. Boorman and Ronald L. Breiger. *Social Structure from Multiple Networks. I. Blockmodels of Roles and Positions.* The University of Chicago Press, 1976.

[19] Jure Leskovec, Jon Kleinberg, Christos Faloutsos. *Graphs over Time: Densification Laws, Shrinking Diameters and Possible Explanations.* 2005.

[20] Deepayan Chakrabarti, Ravi Kumar, Andrew Tomkins. *Evolutionary Clustering.* Yahoo! Research, 701 First Ave, Sunnyvale. 2006.

[21] Tanja Falkowski, Jörg Bartelheimer, Myra Spiliopoulou . *Mining and Visualizing the Evolution of Subgroups in Social Networks.* IEEE/WIC/ACM International Conference on Web Intelligence. 2006.

[22] Gergely Palla, Albert-Lászlo Barabási, Tamás Vicsek. *Quantifying social group evolution.* Nature volume 446, pages 664–667. 2007.

[23] Mansoureh Takaffoli, Farzad Sangi, Justin Fagnan, Osmar R. Zaiane. *MODEC — Modeling and Detecting Evolutions of Communities* Department of Computing Science, University of Alberta Edmonton, Alberta, Canada. 2011.

[24] Peter J. Mucha, Thomas Richardson, Kevin Macon, Mason A. Porte,5 Jukka-Pekka Onnela.*Community Structure in Time-Dependent, Multiscale, and Multiplex Networks.*2010.

[25] Mansoureh Takaffoli, Farzad Sangi, Justin Fagnan, Osmar R. Zaiane. *A Framework for Analyzing Dynamic Social Networks* Department of Computing Science, University of Alberta, Edmonton, Alberta, Canada. 2010.

# Přílohy na CD

A) Text v elektronické podobě

   - \Diplomová práce\text\DP_BUC0022.pdf

B) Zdrojový kód aplikace pro detekci a evoluci komunit

   - \Diplomová práce\src\

C) Spustitelný soubor aplikace pro detekci a evoluci komunit

   - \Diplomová práce\app\CommunityEvolution.exe

D) Použité datové kolekce

   - \Diplomová práce\data\